İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

BUILDING an AUTONOMOUS WHEELED ROBOT and MAPPING

M.Sc. Thesis by

Muhittin ATİLLA, B.Sc.

Department : Mechatronics Engineering

Programme: Mechatronics Engineering

JUNE 2007

İSTANBUL TECHNICAL UNIVERSITY ★ INSTITUTE OF SCIENCE AND TECHNOLOGY

BUILDING an AUTONOMOUS WHEELED ROBOT and MAPPING

M.Sc. Thesis by

Muhittin ATİLLA, B.Sc.

(518041013)

Date of submission : 15 May 2007

Date of defence examination: 13 June 2007

Supervisor (Chairman): Asst.Prof.Dr. Levent OVACIK (İTÜ.)

Members of the Examining Committee Prof.Dr. Sait TÜRKÖZ (İTÜ.)

Assoc.Prof.Dr. Şeniz ERTUĞRUL (İTÜ.)

JUNE 2007

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ FEN BİLİMLERİ ENSTİTÜSÜ

TEKERLEKLİ OTONOM ROBOT YAPIMI ve HARİTALAMA

YÜKSEK LİSANS TEZİ

Müh. Muhittin ATİLLA
(518041013)

Tezin Enstitüye Verildiği Tarih : 15 Mayıs 2007
Tezin Savunulduğu Tarih : 13 Haziran 2007

Tez Danışmanı : Yrd.Doç.Dr. Levent OVACIK (İTÜ.)

Diğer Jüri Üyeleri Prof.Dr. Sait TÜRKÖZ (İTÜ.)

Doç.Dr. Şeniz ERTUĞRUL (İTÜ.)

HAZİRAN 2007

FOREWORD

People have always dreamed of building intelligent machines to perform tasks. Today, these machines are called Robots. Application areas of the Robots vary from factory floors to entertainment, toys, personal service, medicine, surgery, industrial automation, hazardous environments (space, underwater). Regardless of the form of the robot or the task it must perform, robot must maneuver through our world. Most mobile robots follow fixed paths either through using inductive sensor to locate a buried wire or a simple optical sensor to follow a white line. They have very limited sensing capabilities. In the face of obstacles, they have no option but to stop. Many areas in which mobile robots could make an essential contribution, such as nuclear plant, oil rigs, and surveillance demand greater flexibility and a degree of freedom from the environment. In all of these applications mobile robots may be required to explore and map environments, and make changes to the predetermined plans to cope with uncertain conditions. Therefore, mobile robot localization and mapping, the process of simultaneously tracking the position of a mobile robot relative to its environment and building a map of the environment, has been a central research topic in robotics over the past years.

In the scope of this thesis, I have tried to develop an autonomous wheeled mobile robot that will map its environment and locate itself on the map. I would like to thank to my advisor Asst. Prof. Levent Ovacık for his advices and thank to my mother and father Safiye & Rıfkı ATILLA respectively for their support and patience.

Muhittin ATILLA

JUNE 2007

CONTENTS

ABBREVIATIONS

PWM : Pulse Width Modulation
CCP : Capture Compare PWM
MCU : Micro Controller Unit
PIC : Programmable Interface Controller
USB : Universal Serial Bus
RF : Radio Frequency
MSSP : Master Synchronous Serial Port
PSP : Paralel Slave Port
I2C : Inter Integrated Circuit
SCL : Serial Clock
SDA : Serial Data
SPI : Serial Peripheral Interface
USART : Universal Synchronous Asynchronous Receiver Transmitter
I/O : Input / Output
SLAM : Simultaneous Localization and Mapping
MCL : Monte Carlo Localization
DR : Dead Reckoning
EM : Expectation Maximization

LIST OF TABLES

LIST OF FIGURES

TEKERLEKLİ OTONOM ROBOT YAPIMI VE HARİTALAMA

ÖZET

Bu tez kapsamında, içinde bulunduğu ortamın haritasını çıkarabilme ve bu ortam içinde kendini konumlandırabilme yeteneklerine sahip robotik bir sistem geliştirilmiştir. Bu sistem tekerlekli robot ve masaüstü bilgisayar olmak üzere başlıca iki ana kısımdan oluşmaktadır. Hareketli robot masaüstü bilgisayara kablosuz radyo frekans hattı ile bağlanmakta, böylece iş hacmi yüksek işlemler robotun üzerindeki mikroişlemciye nazaran daha hızlı bir işlemciye sahip masaüstü bilgisayar üzerine transfer edilmekte ve ayrıca kullanıcıya robotu uzaktan kumanda etme imkanı verilmektedir. Bu sistemin çıktısı, kullanıcının bilgisayar ekranından görebileceği robotun içinde bulunduğu ortamın haritasıdır.

Robotun keşif ve harita çıkarma yeteneği büyük ölçüde tasarımında kullanılan algılayıcı ve eyleyicilere bağlıdır. Robotun tasarımında, ortam algılama, yön ölçme ve dolaşım gibi bir çok amaca hizmet eden ucuz, küçük fakat güvenilir algılayıcı ve eyleyiciler seçilmiştir. Ortam haritasının çıkarılmasında ultrasonik mesafe ölçer ve dijital pusula kullanılmıştır. Bununla birlikte robotun konumunun takibinde kullanılmak üzere tekerleklerine optik enkoderler yerleştirilmiştir. Algılayıcı ve eyleyicilerin seçiminde boyutları, doğrulukları ve mikroişlemciye olan bağlantıları dikkate alınmıştır. Algılayıcılar tarafından ölçülen tüm değerler, en düşük seviyede PIC18F452(Microchip) mikroişlemcisi tarafından alınıp işlenmekte ve daha karmaşık hesaplama ve bilgi depolamaya olanak tanıyan masaüstü bilgisayara kablosuz radyo frekans hattı ile aktarılmaktadır.

Bu çalışmanın ikinci bölümünde, tekerlekli robotun tasarım ve montajı üzerinde kısaca durulmuştur. Önce robotun mekanik inşası incelenmiş daha sonra keşif ve haritalama için robot üzerine monte edilen algılayıcı ve eyleyiciler'e değinilmiştir. Bölümün devamında, sırasıyla robotun kumanda sistemi ve programlanması açıklanmıştır. Üçüncü bölümde, robotik konumlandırma ve haritalama teknolojileri üzerinde durulmuştur. Bu bölümün başlangıcında, konumlandırma ve haritalama üzerini yapılan çalışmalar kısaca özetlenmiştir. Daha sonra konumlandırma ve haritalama uygulamalarında kullanılan bir çok yöntem açıklanmıştır. Dördüncü bölümde, bu çalışmada kullanılan robotik konumlandırma ve haritalama algoritmaları açıklanmıştır.

BUILDING an AUTONOMOUS WHEELED ROBOT and MAPPING

SUMMARY

In the scope of this thesis, a robotic system that is capable of mapping an area of its environment and locating itself within the environment is developed. The system is mainly comprised of two parts: a wheeled mobile robot and a remote computer. The mobile robot connects to a remote computer through the wireless RF link which enables transferring majority of the processing burden to the faster computer and also allows the user to control the robot from this remote workstation. The output of the system is the map which the user is able to see on the computer screen.

The exploration and map-building abilities of a robot are strongly dependent of its sensors and actuators. Inexpensive, small but reliable sensors and actuators are selected for a number of purposes like environment sensing, orientation measurement and navigation. An ultrasonic range finder and a digital compass are used to map the area and orient the robot within it. Furthermore, each wheel on the robot has an optical encoder which enables the robot to keep track of the robot location. When choosing sensors and actuators the size, accuracy and electronics interface of them are especially taken into account. All the data acquired from the sensors are taken and processed by the Microchip's PIC18F452 microcontroller on the lowest level and sent to the remote computer through wireless RF link for running complex calculation, storing data and interfacing with a user.

In the second chapter of this work, design and assembly of the mobile robot are briefly described. First mechanical construction of the mobile robot is explained. Then, sensors and actuators mounted on the mobile robot for exploration and map building purposes are investigated. In the remaining sections of this chapter, the mobile robot's control system and programming are described respectively. Third chapter is dedicated to the localization and mapping theory. In the beginning of this chapter, historical overview of the robotic mapping and general concepts are briefly discussed. Then a variety of methods used in robotic mapping application are explained. In the fourth chapter, the localization and mapping algorithms used in this study are described.

1. INTRODUCTION

Robotics is the science of sensing and manipulating the physical world through computer controlled devices. Examples of successful robotic systems include mobile platforms for planetary exploration, industrial robotics arms in assembly lines, cars that travel by themselves and manipulators that assist surgeons. Robot can take on many forms and constructing them involves addressing many challenges in engineering computer science, mechanical science, electronics and so on. Regardless of the form of the robot or the task it must perform robot must maneuver through our world. In almost all of robotic applications, mobile robot may be required to explore and map the environments. Therefore, the problem of robotic mapping has received considerable attention over the past years. Mapping is the problem of the generating models of robot environments from sensor data. The importance of problem is due to the fact that many successful mobile robot systems require maps for their operation.

In this study, a robotic system that is capable of mapping an area of its environment and locating itself within the environment is developed. The system is mainly comprised of a wheeled mobile robot and a remote computer as shown in Figure 1.1. The mobile robot connects to a remote computer through the wireless RF link which enables transferring majority of the processing burden to the faster computer and also allows the user to control the robot from this remote workstation. The output of the system is the map which the user is able to see on the computer screen.

The exploration and map-building abilities of a robot are strongly dependent of its sensors and actuators. Cheap, small but reliable sensors and actuators are selected for a number of purposes. An ultrasonic range finder and a digital compass are used to map the area and orient the robot within it. Furthermore, each wheel on the robot has an optical encoder which enables the robot to keep track of the robot location. When choosing sensors and actuators the size, accuracy and electronics interface of them are especially taken into account. All the data acquired from the sensors are taken and processed by the Microchip's PIC18F452 microcontroller on the lowest level

and sent to the remote computer through wireless RF link for running complex calculation, storing data and interfacing with a user.

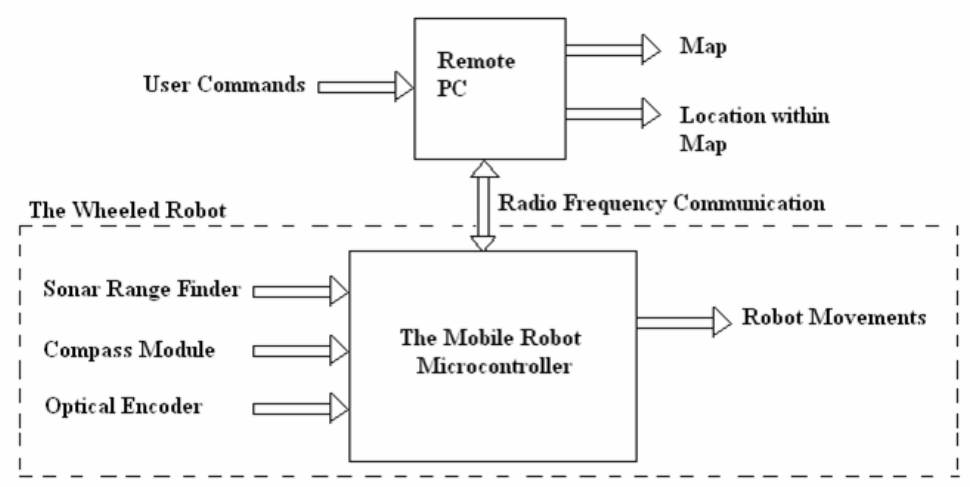

Figure 1.1 : Robotic system description

1.1 Robot Definitions

A robot is a machine that can easily be directed to do a variety of tasks without human supervision. The Robot Institute of America defines a robot in the following manner (1979):

A robot is "A reprogrammable, multifunctional manipulator designed to move material, parts, tools, or specialized devices through various programmed motions for the performance of a variety of tasks" [1].

A more inspiring definition can be found in Webster. According to Webster a robot is: "An automatic device that performs functions normally ascribed to humans or a machine in the form of a human" [2].

1.2 A Brief History of Robots

Although robots are an ancient concept, the word robot was invented in this century. It is derived from the Czechoslovakian word “robota”, meaning “drudgery, servitude, or forced labor. The acclaimed Czech playwright Karel Capek (1890-1938) made the first use of the word ‘robot’ in 1920. Capek used the word “Robot” to describe the

central characters of his classic science fiction play, ”R.U.R.” (Rossum's Universal Robots) [1]. R.U.R's theme, in part, was the dehumanization of man in a technological civilization. The play was an enormous success and productions soon opened throughout Europe and the U.S.

The word "robotics" also comes from science fiction and was first used in Runaround, a short story published in 1942, by Isaac Asimov. This story was later included in Asimov's famous book "I, Robot" published in 1950. Asimov also introduced his three "Laws of Robotics", and he later added a 'zeros law'.

- Law Zero: A robot may not injure humanity, or, through inaction, allow humanity to come to harm.
- Law One: A robot may not injure a human being, or, through inaction, allow a human being to come to harm, unless this would violate a higher order law.
- Law Two: A robot must obey orders given it by human beings, except where such orders would conflict with a higher order law.
- Law Three: A robot must protect its own existence as long as such protection does not conflict with a higher order law [2].

The history of robots expands over 1000 years and continues to modern day robotics. It began in 270 BC when a Greek engineer named Ctesibus made organs and water clocks that contained movable figures. As time progressed, science fiction played an important role in integrating the word "robot" into society.

Today, the robot industry is becoming excessively larger, we have robots such as the: Asimo, and the Atom, both are designed to be capable of human interaction and have a limited level of artificial intelligence. We are still a long way away from having human-like features, such as: eyes, ears, mouth, and skin. Someday through scientific research we will have are humanoid robots, indistinguishable from the humans that created it.

Below are the history timeline of robotics:

- ~270BC -An ancient Greek engineer named Ctesibus made organs and water clocks with movable figures.

- 1818 - Mary Shelley wrote "Frankenstein" which was about a frightening artificial life form created by Dr. Frankenstein.
- 1921 - The term "robot" was first used in a play called "R.U.R." or "Rossum's Universal Robots" by the Czech writer Karel Capek. The plot was simple: man makes robot then robot kills man!
- 1941 - Science fiction writer Isaac Asimov first used the word "robotics" to describe the technology of robots and predicted the rise of a powerful robot industry.
- 1942 - Asimov wrote "Runaround", a story about robots which contained the "Three Laws of Robotics":
- 1948 - "Cybernetics", an influence on artificial intelligence research was published by Norbert Wiener
- 1956 - George Devol and Joseph Engelberger formed the world's first robot company.
- 1959 - Computer-assisted manufacturing was demonstrated at the Servomechanisms Lab at MIT.
- 1961 - The first industrial robot was online in a General Motors automobile factory in New Jersey. It was called UNIMATE.
- 1963 - The first artificial robotic arm to be controlled by a computer was designed. The Rancho Arm was designed as a tool for the handicapped and its six joints gave it the flexibility of a human arm.
- 1965 - DENDRAL was the first expert system or program designed to execute the accumulated knowledge of subject experts.
- 1968 - The octopus-like Tentacle Arm was developed by Marvin Minsky.
- 1969 - The Stanford Arm was the first electrically powered, computer-controlled robot arm.
- 1970 - Shakey was introduced as the first mobile robot controlled by artificial intelligence. It was produced by SRI International.
- 1974 - A robotic arm (the Silver Arm) that performed small-parts assembly using feedback from touch and pressure sensors was designed.
- 1979 - The Standford Cart crossed a chair-filled room without human assistance. The cart had a TV camera mounted on a rail which took pictures from multiple angles and relayed them to a computer. The computer analyzed the distance between the cart and the obstacles. [3]

1.3 Application Areas of Robots

Robots come in many shapes and sizes and have many different abilities. Robots are moving away from factory floors to entertainment, toys, personal service, medicine, and surgery, military and hazardous environments (space, underwater).

1.3.1 Industry

When doing a job, robots can do many things faster than humans. Robots do not need to be paid, eat, drink, or go to the bathroom like people. They can do repetitive work that is absolutely boring to people and they will not stop, slow down, or fall to sleep like a human.

Figure 1.2 : Panasonic the VR-GII 6 axis industrial robot [4]

1.3.2 Entertainment

At first, robots where just for entertainment, but as better technology became available, real robots were created. Many robots are still seen on T.V. (Star Trek - The Next Generation) and in the movies (The Day the Earth Stood Still, Forbidden Planet, Lost in Space, Blade Runner, Star Wars). These imaginary robots do a lot of things that the real ones can not do. Some robots in movies are made to attack people, but in real life they cannot really hurt people at all because they are not in control of themselves.

Figure 1.3 : Famous robots from “Star Wars” movie

1.3.3 Toys

The new robot technology is making interesting types of toys that children will like to play with. One new robotic toy is the "Furby", which became available in stores for Christmas 1998 and continues to be very popular. Another is the "Lego Mindstorms" robot construction kit. These kits, which were developed by the Lego Company with M.I.T. scientists, let kids create and program their own robots. A third is "Aibo", Sony Corporation's robotic dog.

Figure 1.4 : "Aibo", Sony Corporation's robotic dog

1.3.4 Medicine

Sometimes when operating, doctors have to use a robot instead. A human would not be able to make a hole exactly one 100th of a cm wide and long. When making medicines, robots can do the job much faster and more accurately than a human

can. Also, a robot can be more delicate than a human. Where extreme precision and delicacy is necessary, and the margin for error slim, robots are becoming increasingly important. Robots are performing heart surgery without opening patients' chests, pregnant robots are testing medical students, and others are checking medical prescriptions for errors.

1.3.5 Military and Police

Police need certain types of robots for bomb-disposal and for bringing video cameras and microphones into dangerous areas, where a human policeman might get hurt or killed. The military also uses robots for locating and destroying mines on land and in water, entering enemy bases to gather information and spying on enemy troops.

Figure 1.5 : The Remotec F6A from Nothrop Grumman

1.3.7 Exploration

People are interested in places that are sometimes full of danger, like outer space, or the deep ocean. But when they can not go there themselves, they make robots that can go there. The robots are able to carry cameras and other instruments so that they can collect information and send it back to their human operators. The below picture shows the "Sojourner" micro rover robot being repaired at the Nasa's Jet Propulsion Labs. Sojourner landed on the surface of Mars on July 4, 1998.

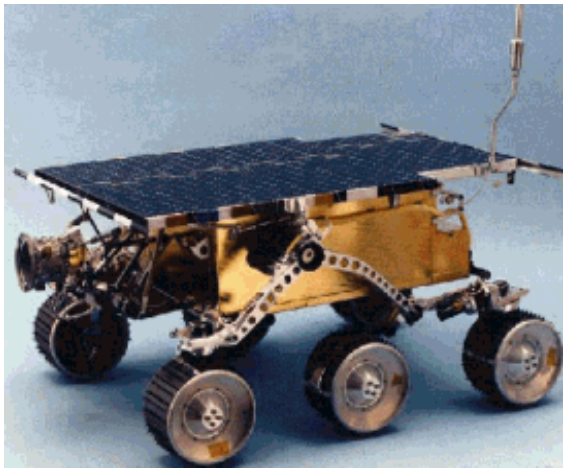

Figure 1.6 : The Sojourner, a 6-wheeled vehicle [5]

2. DESIGN AND ASSEMBLY OF THE WHEELED ROBOT

Building and programming a robot is a combination of mechanics, electronics, computer science, communication and lots of problem solving. At a very basic level, a robot consists of :

- A mechanical device, such as a wheeled platform, arm, or other construction capable of interacting with its environment.
- Sensors on or around the device that are able to sense the environment and give useful feedback to the device.
- Systems that process sensory input in the context's of device current situation and instruct the device to perform actions in response.

In this section, the mechanical construction of the robot based on the design in [6] will be first explained, then electronics components on the robot and integration of each ones will be dealt with, respectively. Finally, programming and testing of complete wheeled robot will be detailed. The complete wheeled robot designed in this project is shown in Figure 2.1.

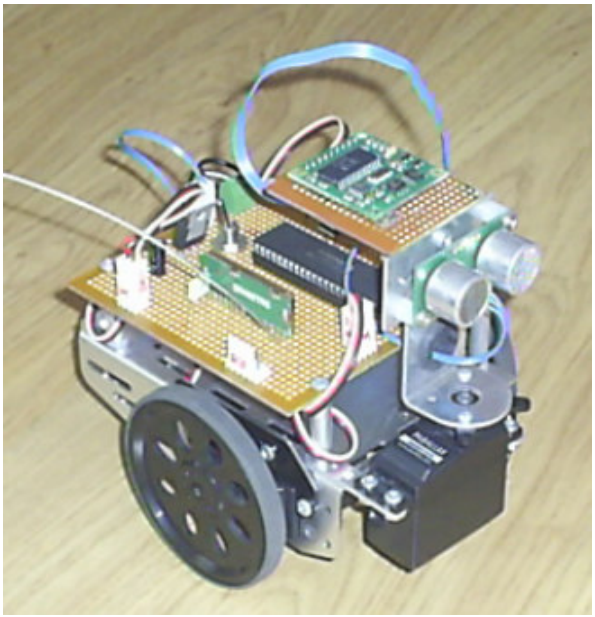

Figure 2.1 : The wheeled mobile robot

2.1 Mechanical Construction

All of the tools used in building of the robot are common and can be found in most households and school shops. They can also be purchased at local hardware stores.

Assembly Tools:

- Screwdrivers
- various wrenches
- needle-nose and general pliers
- handheld electric drill and drill bit set
- work bench vise
- sturdy wire cutter
- hammer
- saw
- cutter

Figure 2.2 : Assembly tools

The mechanical architecture of the robot body are composed of the following parts as shown in Figure 2.3. Without some type of body, a robot isn't a robot at all but some form of artficial intelligence.

Mechanical Parts:

- 1 pc. Metal Chassis
- 4 pcs. 25.4mm Standoff
- 4 pcs. Pan Head Screws, 6.3mm 4-40,
- 1 pc. Rubber Grommet, 10.3mm
- 2 pcs. Continuous Rotation Servos
- 8 pcs. Pan Head Screws, 9.5mm 4-40

- 8 pcs. Nuts, 4-40
- 1 pc. Battery Pack including 2 pc. Screws 9.5mm 4-40 and Nuts 4-40
- 2 pcs. Plastic Machined wheels and 2pc. Screws
- 1 pc. Ball Tail Wheel
- 2 pcs. Rubber Band Tire
- 1 pc. 1.6mm Cotter Pin
- 2 pcs. Mounting Plates and 2 pc. Hex. Nuts 4-40 for encoder

Figure 2.3 : Mechanical parts

2.1.1 Mounting the Top Side Hardware

In this first part of the robot assembly, below listed mechanical parts are used as indicated in Figure 2.4 to build the Robot's Top Side Hardware.

Part List:

- 1 pc. Metal Chassis
- 4 pcs. 25.4mm Standoff
- 4 pcs. Pan Head Screws, 6.35mm 4-40
- 1 pc. Rubber Grommet, 10.3mm

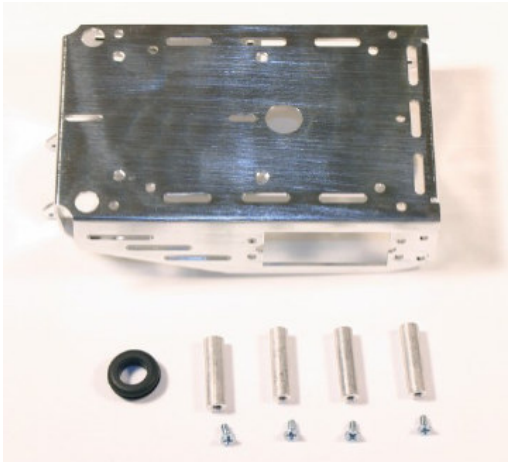

Figure 2.4 : Mounting accessories for the robot body

Below instructions are followed step by step in order for building The Top Side Hardware.

- Insert the 10mm rubber grommet into the hole in the center of the Metal chassis.
- Make sure the groove in the outer edge of the rubber grommet is seated on the edge of the hole in the chassis.
- Use the four 6.3mm 4-40 screws to attach the four standoffs to the chassis as shown in Figure 2.5.

Figure 2.5 shows the resulted Top Side Hardware after having completed above instructions.

Figure 2.5 : Top side hardware

2.1.2 Mounting the Servos on the Chassis

In this part of the robot assembly, below listed mechanical parts are used as illustrated in Figure 2.6 to be able to assemble the two servo motors of the right and left wheels of the robot to the top chassis.

Part List:

- 2 pcs. Continuous Rotation Servos (for right and left wheel)
- 8 pcs. Pan Head Screws, 9.5mm 4-40
- 8 pcs. Nuts, 4-40

After having collected above part list, the instructions written below are followed step by step in order to mount the servo motors to the chassis.

Figure 2.6 : Servo motors and accessories

Instructions:

- Remove the servo horn (four pronged plastic piece on the end of the servo shaft) from each servo by removing the screw and pulling the horn off as indicated in Figure 2.7. The servo horns are spare parts that is not used, but the screws will be used later to attach the wheels.

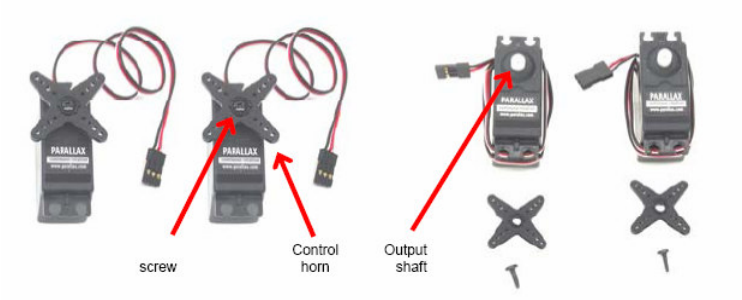

Figure 2.7 : Removing servo horns

- Set the robot chassis on its edge then slide a servo into the servo mounting hole from above, positioning the servo such that the shaft is nearest to the center of the chassis, as shown in Figure 2.8.

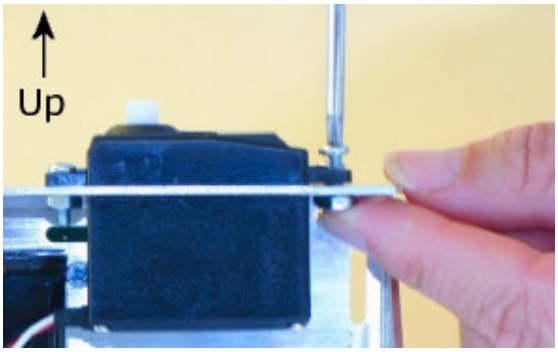

Figure 2.8 : Installing the servo motor

- With the chassis on its edge, fasten the servo in place with four long round head screws and nuts, positioning each nut from below using my finger and inserting the screws from the above, as shown in Figure 2.8. This will be easiest if the chassis is kept on its edge and the nut is positioned using finger, under the hole before inserting the screw.
- Repeat the previous steps to install the other servo.

After having completed the above steps, the partially assembled robot chasses together with the mounted servo motors is as shown in Figure 2.9.

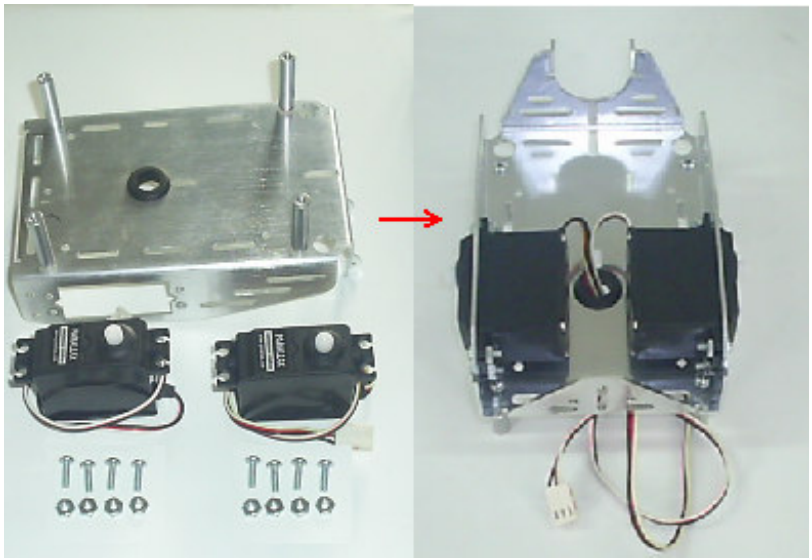

Figure 2.9 : Partially assembled robot chassis

2.1.3 Mounting the Battery Pack

The Battery pack as shown in Figure 2.10 is used for housing of four AA sizes batteries. The items on the below part list are used for the installation of the battery compartment.

Part List:

- 1 pc. Battery Pack
- 2 pcs. Screws 9.5mm 4-40
- 2 pcs. Nuts 4-40

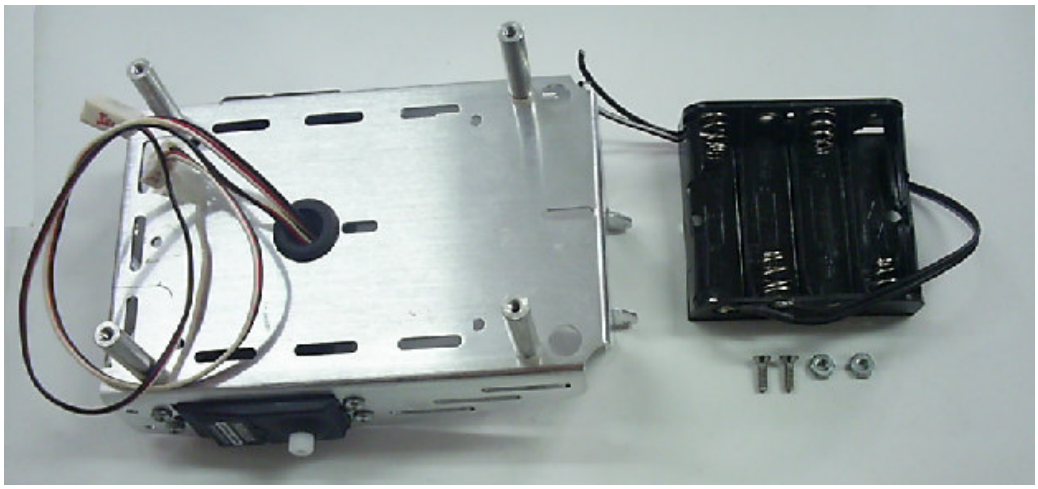

Figure 2.10: Robot body and battery pack

Below instructions are followed during mounting of battery pack to the robot chasses.

- Use the flathead screws and nuts to attach the battery pack to underside of the Robot chassis.
- Make sure to insert the screws through the battery pack, then tighten down the nuts on the topside of the chassis.
- As shown on the right side of Figure 2.11, pull the battery pack's power cord through the hole with the rubber grommet in the center of the chassis.
- Pull the servo lines through the same hole.
- Arrange the servo lines and supply cable as shown.

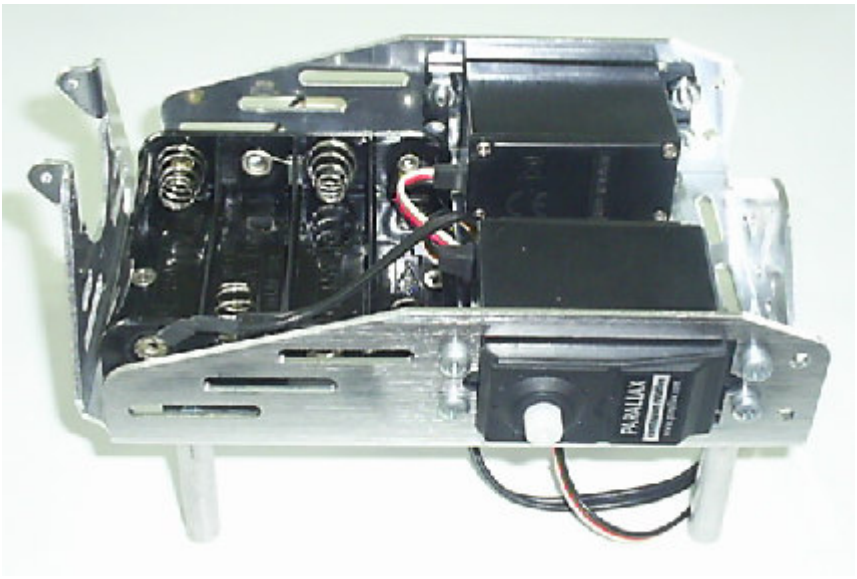

Figure 2.11 : Partially assembled robot chassis

2.1.4 Mounting the Wheels

Figure 2.12 shows the wheels and its mounting accessories. The diameter of the wheel is 67mm and it has 8 holes for position tracking by using encoder.

Part List:

- 2 pcs. Plastic Machined Wheels and 2 pc. Screws
- 1 pc. Ball Tail Wheel
- 2 pcs. Rubber Band Tire
- 1 pc. 1.6mm Cotter Pin

Figure 2.12 : The wheels and its mounting accessories

Instructions followed during assembly:

- Stretch a rubber band tire over each large wheel.
- Press a large wheel on to a servo shaft.
- Fasten the wheel in place with the black screw you previously removed from the servo.
- Repeat the previous steps for the other large wheel.
- Hold the tail wheel ball between the mounts on the chassis then slide the cotter pin through the mounting holes and the hole in the ball.
- Bend the ends of the cotter pin apart so it can't slide out of place as shown in Figure 2.13.

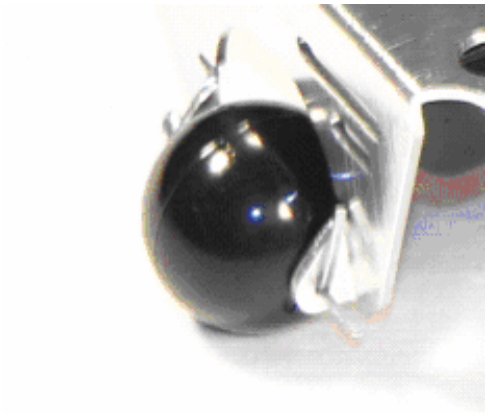

Figure 2.13 : Robot's tail wheel

The right side of Figure 2.14 shows the Robot's tail wheel mounted on the chassis. The tail wheel is merely a plastic ball with a hole through the center. The left side of Figure 2.14 shows Robot's drive wheels mounted on the servos.

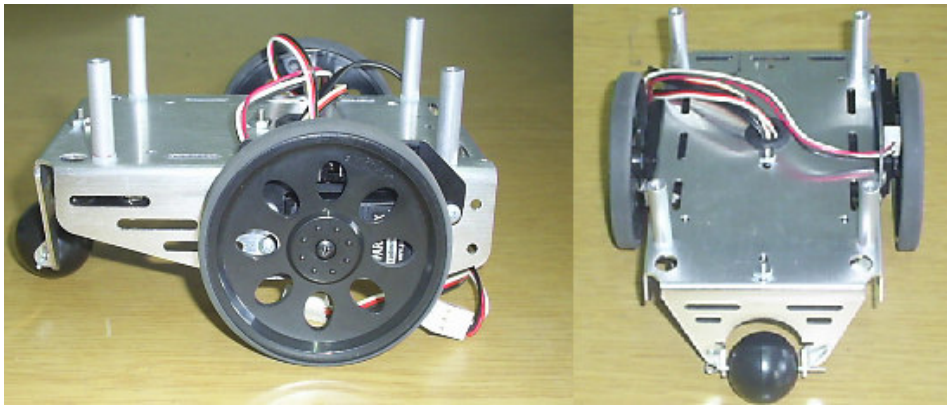

Figure 2.14 : Robot's wheel

2.1.5 Mounting the Optical Encoder

Figure 2.15 indicates the optical encoder and its mounting components listed below.

Part List:

- 2 pcs. Reflective Sensor
- 2 pcs. Mounting Plates
- 2 pcs. 6.3mm 4-40 Screws and 4-40 Hex. Nuts

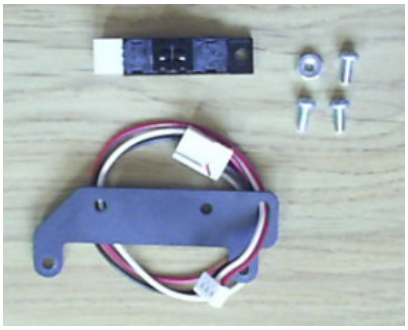

Figure 2.15 : Optical encoder and its mounting accessories

Below instructions are followed to be able to assemble the encoder to the robot chassis :

- Remove both wheels from the robot and the top two screws holding each servo. Loosen the remaining servo mounting screws.
- Take one of the mounting plates, and thread the wires from one of the cables through the oblong hole. Insert the sensor's alignment pin into the mating mounting plate hole. Secure the sensor to the mounting plate with a screw through the other set of holes and a nut on the back.
- Repeat for the other sensor, but with the opposite orientation. When finished, there are two mounted sensors that are mirrors of each other as shown in Figure 2.16.

Figure 2.16 : Mounted encoders

- Using the servo screws removed in the first step, mount each sensor to the chassis so the lenses are near to the servo shaft as shown in Figure 2.17. Do not tighten the screws yet.

Figure 2.17 : Assembled optical encoder

- Viewing the servos from the chassis bottom, adjust their positions so that their shafts align as closely as possible. Either center both in their respective cutouts, or push them both toward (not away from) the sensor connector. Just make sure they're the same. Now tighten all eight servo screws.

2.1.6 Assembling Range Finder and Compass Module

These mounting brackets of Range Finder and Compass Module are made of aluminum and are mounted on the shaft of servo motor located in front of the Robot. The functions of these brackets are to carry the ultrasonic range finder and the compass module on them. Since they are mounted on the shaft of the servo motor, these brackets can be easily rotated left and right. Mounting accessories used in assembly are listed below.

Part List:

- 11 pcs. 4/40 nuts, zinc plated
- 3 pcs. Nylon washer (size 4)
- 1 pc. Rubber grommet – 10.3mm size
- 7 pcs. 4/40 6.3mm long pan head Phillips screw
- 2 pcs. 2/56 6.3mm long pan head Phillips screw
- 4 pcs. 4/40 12.7mm long pan head Phillips screw
- 2 pcs. 2/56 nut
- 2 pcs. Nylon spacer 6.3mm long (size 4)

- 2 pcs. Straight brackets
- 1 pc. mounting bracket of Ultrasonic Range Finder
- 1 pc. Mounting Bracket of Compass Module
- 1 pc. Servo extension cable 25cm
- 1 pc. Parallax Standard Servo

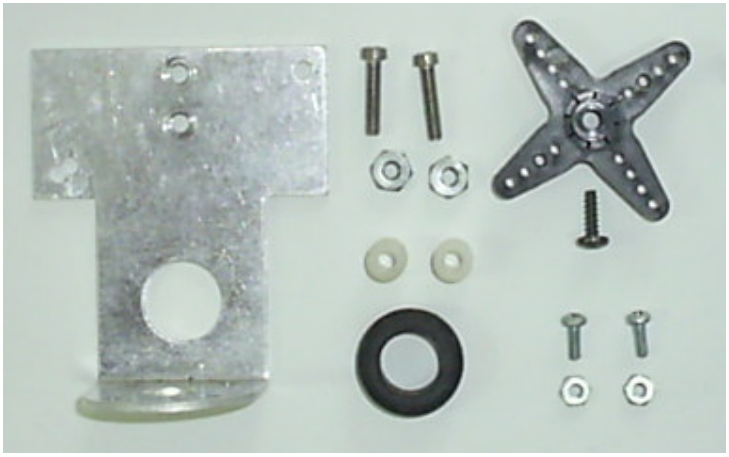

Figure 2.18 : Mounting bracket and accessories for the ultrasonic range finder

The following instructions are followed step by step to assemble the Range Finder's and Compass Module's Mounting Brackets.

Instructions:

- Enlarge two holes on the standard servo horn shown in Figure 2.18 with a 2.0mm drill bit so the screw can make its own threads by taking into account that the servo horn plastic is quite brittle and can crack if it is enlarged with a screw.

Figure 2.19 : Attaching mounting bracket and servo horn

- Attach Mounting Bracket of The Ultrasonic Range Finder to the servo horn using (2) 2/56 6.3mm long screws and nuts. Put the rubber grommet in the bracket's larger hole as shown in Figure 2.19. In the next step, the mounting bracket which will carry the Compass Module will be attached to this mounting bracket.

- Assemble Mounting Bracket of the Compass Module shown in Figure 2.20 to the Mounting Bracket of Ultrasonic Range Finder mounted in the previous step.

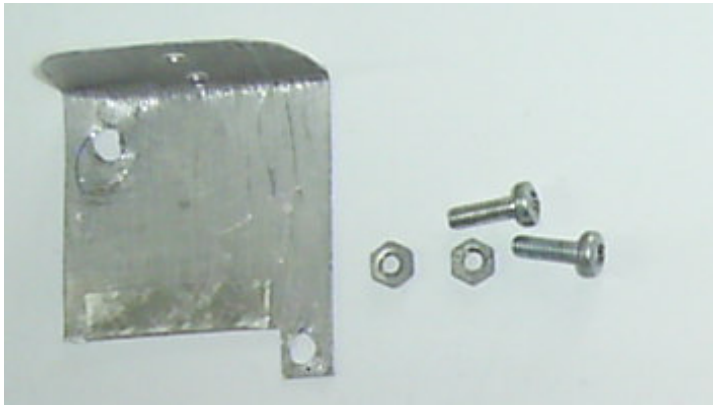

Figure 2.20 : Mounting bracket and accessories for the compass module

Figure 2.21 shows the resulted assembly. This construction will carry both ultrasonic range finder and compass modules on it.

Figure 2.21: Resulted assembly for the sensors' housing

- Attach the ultrasonic range finder shown in Figure 2.22 to the Mounting Bracket using (2) 4/40 12mm long screws, (2) 6mm long nylon spacers and (2) 4/40 nuts.

Figure 2.22 : The ultrasonic range finder

Figure 2.23 shows the resulted assembly after having attached the ultrasonic range finder.

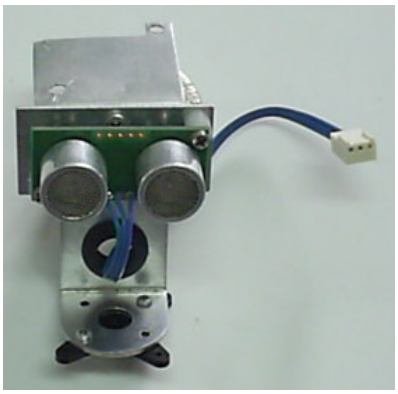

Figure 2.23 : Mounting the range finder

- Attach the compass module shown in Figure 2.24 to the on top of its Mounting Bracket.

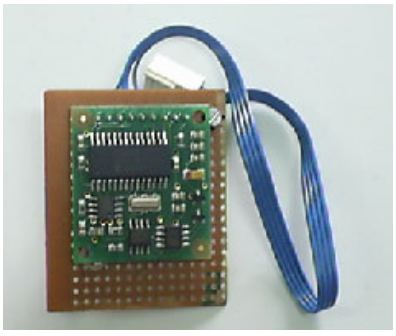

Figure 2.24 : The compass module

Resulted assembly is indicated in figure 2.25 after having mounted compass module to the previous assemble.

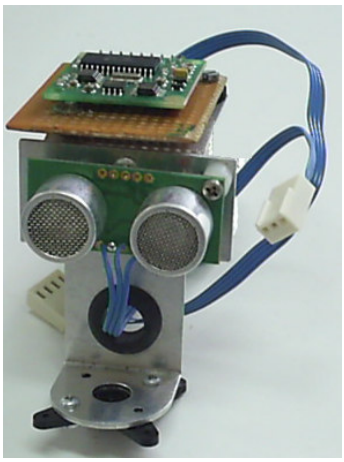

Figure 2.25 : Resulted assembly after sensors' mounting

- Front of the top side hardware already assembled in section 2.1.1 is used for housing the servo motor used to carry and rotate the resulted assembly in Figure 2.25. Figure 2.26 shows the servo motor and its mounting accessories. The chassis's front right mounting position is used for one of the straight bracket, the other straight bracket is mounted on one of the chassis's front slots using a 4/40 6.3mm screw and nut. Three nylon washers are installed under the other three standoffs so that it remains level.

Figure 2.26 : Servo motors and its mounting accessories

- Attach servo motor to the straight brackets using (4) 4/40 screws and (4) 4/40 nuts. Replace the servo's shaft screw once the Mounting Bracket is installed. Figure 2.27 shows the completed wheeled robot.

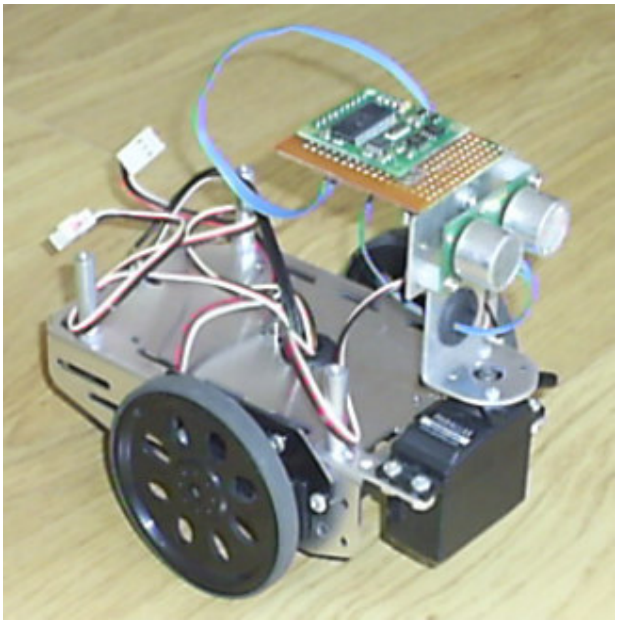

Figure 2.27 : The completed wheeled mobile robot

2.2 Actuators and Sensors of the Wheeled Robot

After having completed the mechanical construction of the robot, this section will deal with actuators and sensors mounted on the wheeled mobile robot. Continuous rotation and standard servos are used as an actuator to drive the wheels and head of the robot respectively. The robot can be programmed to perform a variety of maneuvers such as forward, backward, rotate left, rotate right, and pivoting turns. All these movements are achieved by continuous rotation servo motors that make the wheels of the robot turn. The robot can also sense its surroundings through a set of sonar 'eyes' attached to the front of the robot. Sonar sensors are excellent sensors to use for mobile robot applications. If the robot needs to navigate through a room filled with obstacles, then it can do it successfully by employing sonar sensors. Moreover, to create a useful local map, the algorithm requires range measurements in a number of different directions. Such measurements are readily obtained by sweeping the sensor. The robot used in this project features a sonar sensor that is mounted on a standard servo motor in front of the robot, so a 180-degree sweep is possible. The robot also has a compass module which has been specifically designed for use in robots as an aid to navigation. The aim is to produce a unique number to represent the direction the robot is facing. The optical encoders are also attached to the wheels

of the robot to keep track of the robot location. By incorporating optical encoders, the robot is able to tell how far each wheel has turned and is, hopefully, able to coordinate the wheels' respective movements to guide to a desired destination. In addition, even without coordinated movement, and given any sequence of encoder outputs (along with the directions of rotation), the robot should be able to tell where it is and in what direction it's pointing. Moreover, the robot includes a wireless communication module for bidirectional data transfer with a remote computer. The system has the remote computer in order for processing the data acquired from the sensors, running complex calculation, storing map data and interfacing with a user. All the data acquired from the sensors is processed by a microcontroller on the robot and transferred to the remote PC by this wireless RF Module.

2.2.1 Continuous Rotation Servo Motor

The Parallax Continuous Rotation servos shown in Figure 2-28 are the motors that will make the Robot's wheels turn. This figure points out the servos' external parts. The Parallax Continuous Rotation servo is ideal for robotic products that need a geared wheel drive or other projects that require a 360º rotation geared motor. Continuous rotation servos are ideal for controlling wheels and pulleys. The servo can be adjusted with a small screw driver if the unit becomes out adjustment on its center set point. The servo is controlled by pulsing of its signal line. Pulse width controls speed and direction of the servo motor.

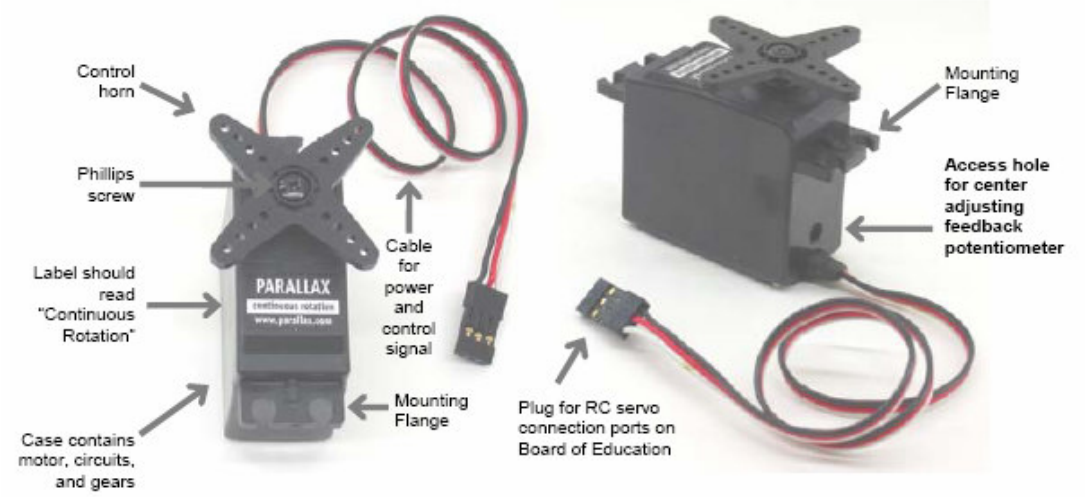

Figure 2.28 : Parallax Continuous Rotation servos

Technical Specifications of the Servo Motor :

- Power 6Vdc max
- Average Speed 60 rpm (with 5Vdc and no torque)
- Weight 45.0 grams
- Torque 3.40 kg-cm
- Size in mm (L × W × H)
  40.5 × 20.0 × 38.0
- Manual adjustment port

2.2.2 Standard Servo Motor

Standard servos are designed to receive electronic signals that tell them what position to hold. In general these servos control the positions of radio controlled airplane flaps, boat rudders, and car steering. In this Project, Parallax standard servo that is ideal for robotics and basic movement projects is used. These servos will allow a movement range of 0 to 180º depending on the pulse width having sent to control line of the servo motor. The robot uses this servo to rotate the mounting brackets of the ultrasonic range finder and compass module mounted on its shaft. In this way, the robot can sense its surroundings through these sensors and servo motor combination.

Technical Specifications of the Standard Servo Motor :

- Power 6Vdc max
- 0º deg to 180º in 1.5 seconds on average
- Weight 45.0 grams
- Torque 3.40 kg-cm
- Size in mm (L × W × H)
  40.5 × 20.0 × 38.0

2.2.3 Ultrasonic Range Finder

The exploration and map-building abilities of a robot are strongly dependent of its sensory skills and its capacity to determine its position. Ultrasonic range sensors provide an inexpensive and reliable means for robot localization and environmental sensing and are commonly used for navigation in unexplored environments. Since

exploration is usually done without having any previous knowledge of the environment, the robot must be able to detect obstacles along its way in order to navigate safely through uncharted areas. While navigating, the robots collect information from their sensors from all possible viewpoints. They allow the robot to avoid obstacles and identify navigable routes. Furthermore, they allow the robot to detect landmarks, which are used for map-building.

The Devantech SRF05 Ultrasonic Range Finder shown in Figure 2.29 is used in this robot.

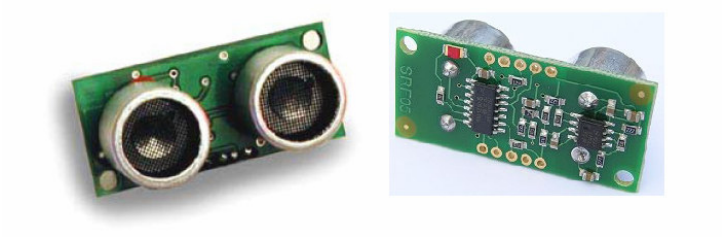

Figure 2.29 : SRF05 ultrasonic range finder

2.2.3.1 How Sonar Works

Figure 2.30 below shows a basic diagram of a generic sonar sensor in operation. An ultrasonic burst of energy is emitted from the transducer. This is known as a ping. The sound waves travel until reflected off of an object. The echoed sound wave then returns to the transducer. The echo may be of smaller amplitude, but the carrier frequency should be the same as the ping. An external timer records the time of flight (the time that the sound waves take to travel to and from the object), which can be converted to distance when considering the speed of sound in air. As the transmitted sound waves propagate from the transducer, they spread over a greater range. In other words, the sound waves propagate from the transducer in the shape of a cone of angle θ.

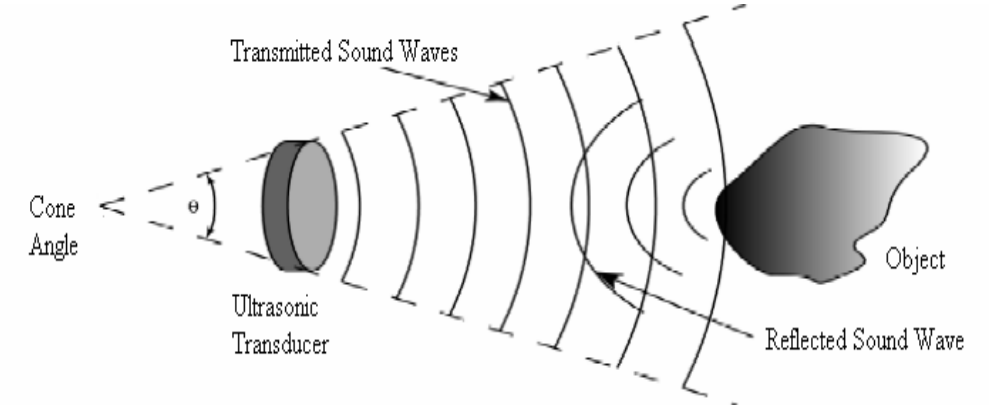

Figure 2.30 : Sonar sensor operation

The Devantech SRF05 Ultrasonic Range Finder is unique in that it has a separate transmitter and receiver. This allows for the smallest minimum detectable distance. The frequency of the ping is 40kHz. The reason for a 40 kHz frequency sound wave is to reduce the chances of false echoes. For example, it is unlikely that a 40 kHz sound wave will come from any other source other than the actual ultrasonic sensor itself. The external timing circuit looks for a 40kHz return signal to identify it as an echo. The SRF05 offers precise ranging information from roughly 1cm to 4 meters. This range and minimal power requirements, 5 volts, make this an ideal ranger for robotics applications.

SRF05 Ultrasonic Range Finder Technical Characteristics :

- Voltage – 5V only required
- Low Current - 4mA Typ.
- Frequency - 40KHz
- Max Range - 4 m
- Min Range - 1 cm
- Modes - Single pin for trig/echo or 2 Pin SRF04 compatible.
- Input Trigger - 10uS Min. TTL level pulse
- Echo Pulse - Positive TTL level signal, width proportional to range.
- Small Size - 43mm × 20mm × 17mm height

2.2.4 Compass Module

The Devantech's CMPS03 Compass Module shown in Figure 2.31 is used in this robot as an aid to navigation. The aim is to produce a unique number to represent the

direction the robot is facing. The compass uses the Philips KMZ51 magnetic field sensor, which is sensitive enough to detect the Earths magnetic field. The output from two of them mounted at right angles to each other is used to compute the direction of the horizontal component of the Earths magnetic field.

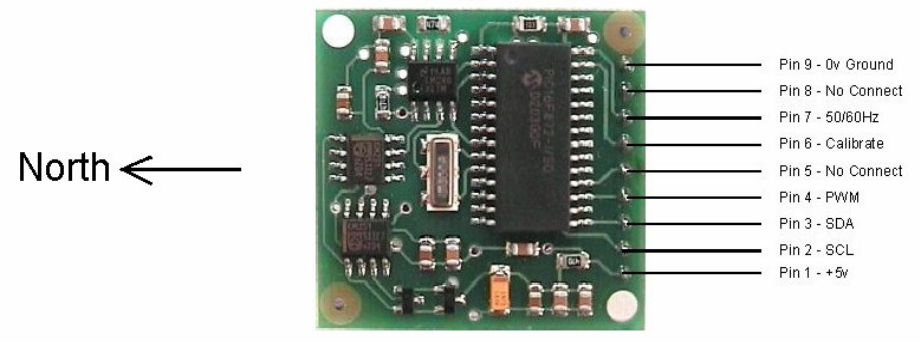

Figure 2.31 : CMPS03 compass module

CMPSO3 Compass Module Technical Characteristics :

- Voltage: 5Vdc only required
- Current: 20mA
- Resolution: 0.1º
- Accuracy: 3-4º approx. after calibration
- Output 1: Timing Pulse 1mS to 37mS in 0.1mS increments
- Output 2: I2C Interface, 0-255 and 0-3599, SCL speed up to 1MHz
- Small Size: 32mm × 35mm

2.2.5 Optical Encoder

The robot created in this project has two wheels which can both move forward or reverse and that they are positioned parallel to one another, and equidistant from the center of the robot. Furthermore, each motor has an optical encoder which enables the robot to keep track of the robot location. By incorporating optical encoders, the robot is able to tell where it is and in what direction it's pointing. The parallax's optical encoder as shown in Figure 2.32 is used in this project.

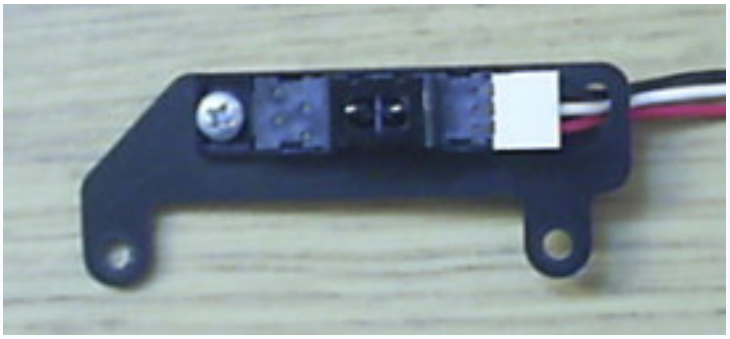

Figure 2.32 : Parallax optical encoder

The sensors emit infrared light and look for its return from a reflective surface. They are calibrated for optimal sensing of surfaces a few millimeters away. The wheels of the Robot, even though they are black, reflect sufficient IR to cause the sensors to respond. When a sensor "sees" part of a wheel, it pulls its output low. When it's looking through a hole, its output floats, and the pullup resistor pulls it high. Because the sensors emit and detect only modulated IR (at about 7.8KHz) they are relatively insensitive to ambient light. As the robot wheel turns, the sensor will see an alternating pattern of hole - no hole - hole -no hole, etc. Its output will be a square wave whose frequency corresponds to the speed of rotation. If the Robot is rolling along the floor, each edge of the square wave will mark an increment of travel slightly more than 26mm, as shown in the illustration in Figure 2.33.

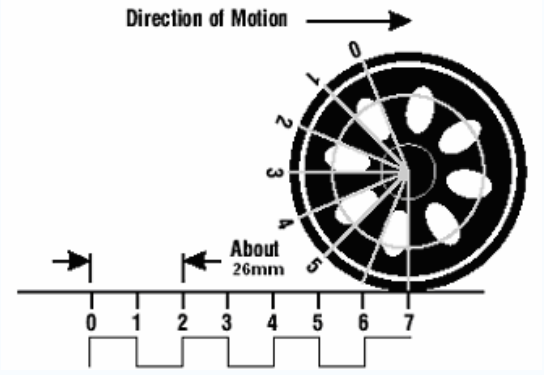

Figure 2.33 : Encoder's sensing distance

By tracking changes in the encoders' outputs, it's possible for an MCU program to tell how far the Robot has traveled. Notice that the encoders themselves do not tell the robot which direction the wheels are turning — only when and how far. But if the robot program is driving the wheel servos, it knows which direction each wheel is turning and can apply this additional information along with the encoder outputs. In

the most sophisticated applications, it is possible not only to keep track of the Robot's position and direction, but also to coordinate the rotations of the two wheels, using the encoders as feedback, to obtain any desired motion contour. But the fact that wheel encoders are never perfect should be taken into account. Uncertainties in the effective wheel diameters can lead to position errors. Further uncertainties in effective wheel spacing during turns can result in direction errors. And even small position and direction errors have a way of accumulating quickly if not periodically corrected using external references.

2.2.6 Radio Frequency Communication Module

The communication between the computer and the Robot is realized by radio frequency. For this purpose, LPRS Easy Radio ER400TRS Transceiver is used by the Robot. The Easy-Radio ER400TRS transceiver incorporates 'Easy-Radio' technology to provide high performance, simple to use radio devices that can transfer data over a range of up to 250 meters Line Of Sight (LOS). ER400TRS module operates at 433-4MHz

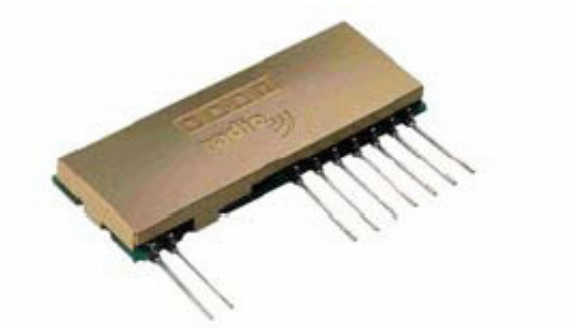

Figure 2.34 : LPRS Easy Radio ER400TRS Transceiver

The Easy-Radio Transceiver is a complete sub-system that combines a high performance very low power RF transceiver, a microcontroller and a voltage regulator as depicted in Figure 2.35. The Serial Data Input and Serial Data Output operate at the standard 19,200 Baud and the two handshake lines provide optional flow control to and from the host. The Easy-Radio Transceiver can accept and transmit up to 180 bytes of data, which it buffers internally before transmitting in an efficient over-air code format.

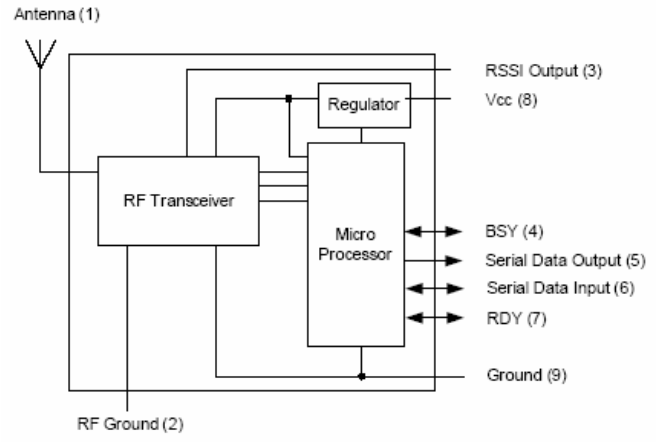

Figure 2.35 : The radio transceiver block diagram

Any other Easy-Radio Transceiver within range that 'hears' the transmission will decode the message and place the recovered data within a receive buffer that can then be unloaded to the receiving host for processing and interpretation. Transmission and reception are bi-directional half duplex i.e. transmit or receive but not simultaneously.

To be able to establish communication between the desktop computer and the robot, the computer also needs to be attached an RF communication module as shown in Figure 2.36. For this purpose, RF04 USB Telemetry module which is the product of Devantech Company is used. The module is powered from the USB bus of the computer, so no external power supply or batteries are required.

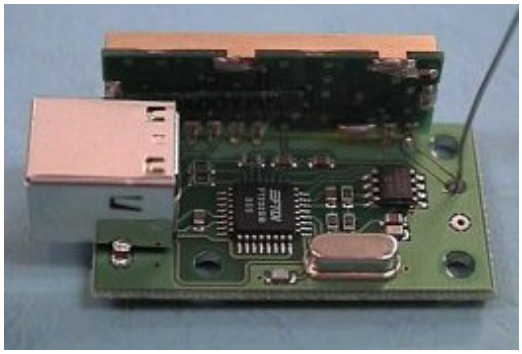

Figure 2.36 : The RF04 USB telemetry module

The heart of this module is the FTDI FT232BM USB chip . Before using the RF04, FTDI's Virtual COM Port Drivers should be installed. These drivers appear to the system as an extra Com Port (in addition to any existing hardware Com Ports).

Application software accesses the USB device in the same way as it would access a standard Windows Com Port using the Windows VCOMM API calls or by using a Com Port Library. The COM port of the computer should be set up for 19200 baud, 8 data bits, no parity and one stop bit.

2.3 Designing the Robot Control System

Today an overwhelming majority of the robots are controlled by microcontrollers (MCUs). Microcontrollers are like the microprocessor (Central processing unit, or CPU) inside large home computers. MCUs are slow and can address less memory than CPUs, but they are designed for real world control problems and are both inexpensive and easy to use. One of the biggest differences between CPUs and MCUs is the number of external components needed to operate them. MCU can often run with zero external parts. Microchip's high performance enhanced flash PIC18F452 microcontroller illustrated in Figure 2.37 is used as the brain of the robot in this project because of the advantages it offered. Before discussing how control circuit is designed for the wheeled robot and how software is developed for it, a basic understanding of what features the Microchip PIC18F452 provides will be discussed. Later, how robot microcontroller is integrated into the mobile robot will be introduced. An important part of this is showing how different sensors, actuators, and peripherals are wired to the microcontroller as well as interfaced to.

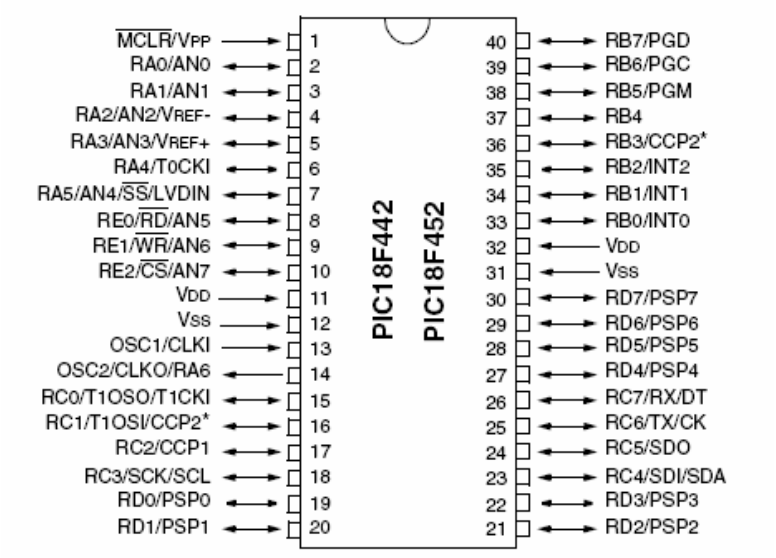

Figure 2.37 : Microchip high performance PIC 18F452 microcontroller

2.3.1 Brain of the Robot

The most important thing is to define requirements that a microcontroller should have before starting to choose what types of microcontroller will be used as a brain of the robot. First of all, since the robot created in this project has sensors and actuators like ultrasonic range finder and servo motor, it should have enough amount of Digital I/O port for interfacing these components to the microcontroller. Internal timers are an important resource for microcontroller and robot applications also. Since the servo motors used on the robot are controlled by the pulse width, the controller to be chosen has to have adequate quantity of timers and interrupt capabilities. As explained in section 2.2.3 , the pulse width of the signal coming from ultrasonic sensor determines the range. The controller needs another timer to be able to extract this range information from the pulse width. In addition, the robot has encoders mounted on its each wheels to keep track of the location. Therefore, at least two counters should be built into the microcontroller to determine exactly how many number of revolutions the wheels has turned. Moreover, the fact that I2C interface is used for getting bearing from the Compass Module requires the controller should support I2C serial communication protocol. Furthermore the data having collected from the sensors of the robot will be sent to the Desktop Computer. This necessities Universal Asynchronous Receiver and Transmitter (UART) feature to be provided by the controller to communicate with the desktop computer.

By taking into account all above features that the robot's microcontroller should have, Microchip High performance PIC18F452 microcontroller is chosen illustrated in Figure 2.38.

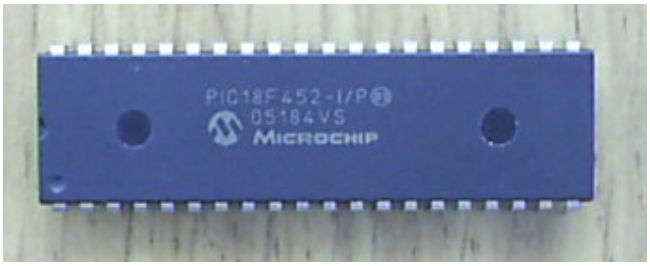

Figure 2.38 : Microchip PIC 18F452

Table 2.1 shows an overview of features that PIC18F452 provides.

Table 2.1 : Microchip PIC18F452 features

Features	PIC18F452
Operating Frequency	DC 40 MHz
Program Memory (Bytes)	32K
Program Memory (Instructions)	16384
Data Memory (Bytes)	1536
Data EEPROM Memory (Bytes)	256
Interrupt Sources	18
I/O Ports	A, B, C, D, E
Timers	4
Capture/Compare/PWM Modules	2
Serial Communications	MSSP, Addressable USART
Parallel Communications	PSP
10-bit Analog-to-Digital Module	8 input channels
RESETS (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST)
Programmable Low Voltage Detect	Yes
Programmable Brown-out Reset	Yes
Instruction Set	75 Instructions
Packages	40-pin DIP 44-pin PLCC 44-pin TQFP

All microcontrollers are built with some kind of I/O capabilities along with some hardware features that are available for use by the application code to simplify the execution of the application. Peripheral Features offered by PIC18F452 are as follows:

- Three external interrupt pins
- Timer0 module: 8-bit/16-bit timer/counter with 8-bit programmable prescaler
- Timer1 module: 16-bit timer/counter
- Timer2 module: 8-bit timer/counter with 8-bit period register (time-base for PWM)
- Timer3 module: 16-bit timer/counter
- Two Capture/Compare/PWM (CCP) modules.
- Master Synchronous Serial Port (MSSP) module, Two modes of operation:
  - 3-wire SPI™ (supports all 4 SPI modes)
  - I2C™ Master and Slave mode
- Addressable USART module:
  - Supports RS-485 and RS-232
- Parallel Slave Port (PSP) module

2.3.2 Power for the Robot

Power for the robot is provided by on-board batteries. The robot has two battery packs: one for motors and one for electronics. The reason for using two battery packs is to minimize the power fluctuations experienced when the motors are turned on and off. For the servo motors, four AA type 1.5V Alcaline batteries were connected together in pack to increase their voltage or capacity as shown in Figure 2.39 .

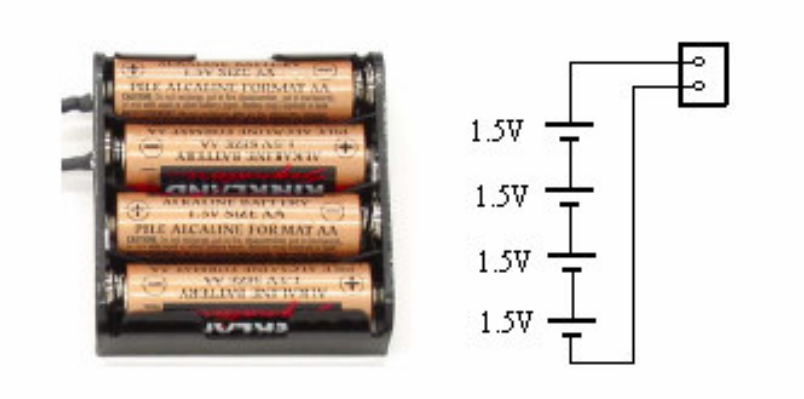

Figure 2.39 : Battery pack for servo motors

As for electronics, they require a particular voltage 5V and want the power to be clean and stable. One rechargeable 9VDC NiMH battery together with a linear voltage regulator LM7805 are used to fulfill these requirements as shown in Figure 2.40.

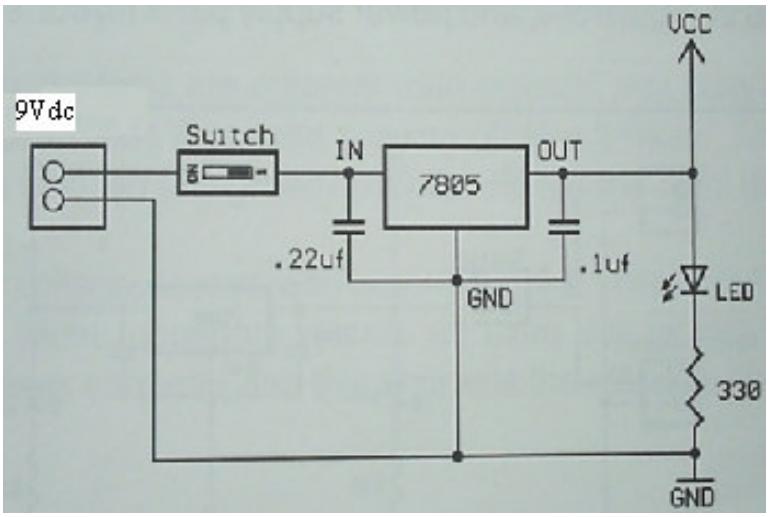

Figure 2.40 : Power supply for the electronics

2.3.3 Oscillator Configuration

High Speed 20MHz Crystal is used as an Oscillator. 20MHz Crystal is connected to the OSC1 and OSC2 pins to establish oscillation. Figure 2.41 shows the pin connections.

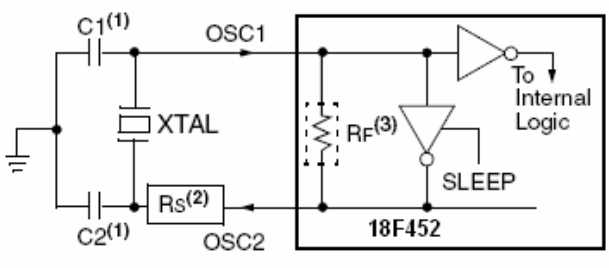

Figure 2.41 : Oscillator connection

2.3.4 Continuous Rotation Servo Motor's Connection

Figure 2.42 shows the connections to the servo motors of the wheels. The digital I/O Port D of the PIC18F452 is used to be able to control the servo motors. RD1 pin 20 and RD2 pin 21 are wired to the control line of the left and right servo motor respectively. Four alkaline AA (1.5 V) batteries are used to power the servo motors.

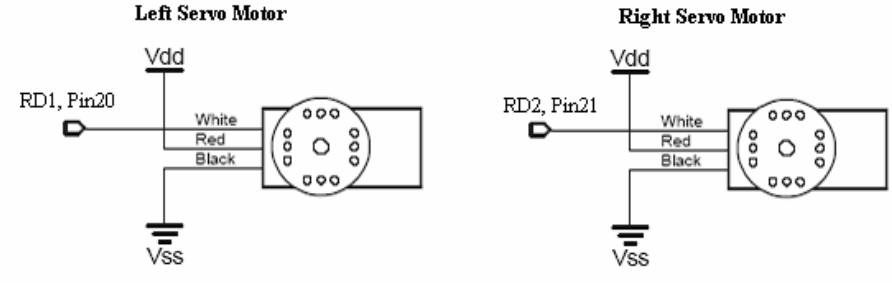

Figure 2.42 : Connections of the continuous rotation servos

2.3.5 SRF05 Ultrasonic Range Finder's Connection

The SRF05 requires three connections to operate. First is the power and ground lines. The SRF05 requires a 5Vdc power supply capable of handling roughly 50mA of continuous output. SRF05 uses a single pin for both Trigger and Echo signals. The echo signal will appear on the same pin as the trigger signal. This pin shall be connected directly to digital I/O ports of a microcontroller. Input/output of the SRF05's control line is connected to the port D's RD5 (pin 28) digital I/O register.

Below, Figure 2.43 depicts the basic schematic of how one should go about connecting the SRF05 to a PIC18F452 microcontroller.

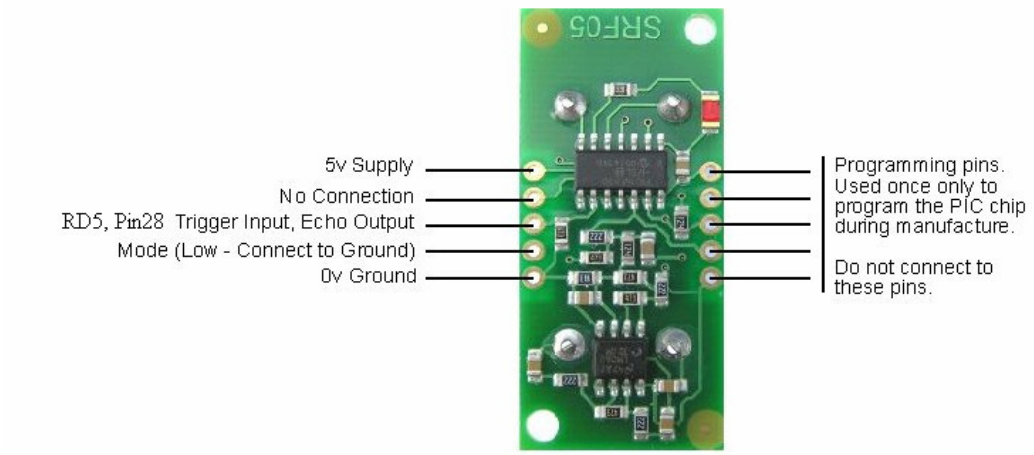

Figure 2.43 : SRF05 ultrasonic range finder connection

2.3.6 CMPS03 Compass Module's Connection

The compass module requires a 5v power supply at a nominal 15mA. There are two ways of getting the bearing from the module. A PWM signal is available on pin 4, or an I2C interface is provided on pins 2, 3. I2C interface is chosen to get the orientation information from the compass module in this project. The PWM signal is a pulse width modulated signal with the positive width of the pulse representing the angle. The pulse width varies from 1mS (0°) to 36.99mS (359.9°). Pin 2 and 3 are used as an I2C interface to get a direct readout of the bearing as shown in Figure 2.44.

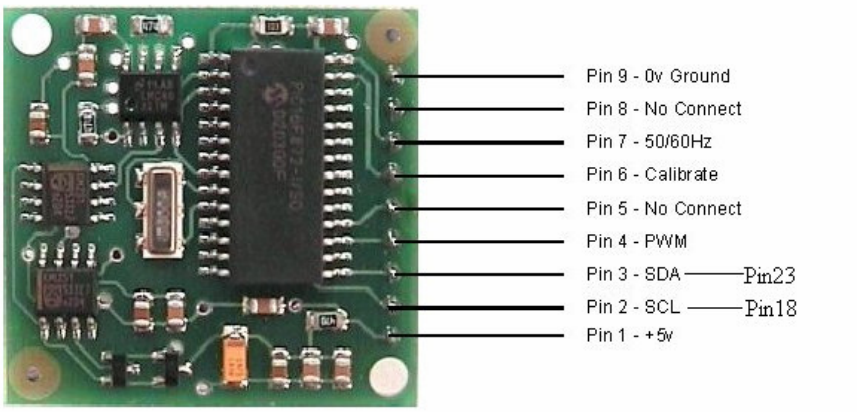

Figure 2.44 : CMPS03 compass module connection diagram

The I2C interface does not have any pull-up resistors on the board, these should be provided elsewhere, most probably with the bus master. They are required on both the SCL and SDA lines, but only once for the whole bus, not on each module.

2.3.7 Optical Encoder Connection

The optical encoder has three wires. These wires are assigned as follows :

Red: Vdd

Yellow: Vss

Black: Signal (open collector)

The encoders of the left and right wheel are connected to the microcontroller's RD5 Pin19 and Pin22 respectively as shown in Figure 2.45.

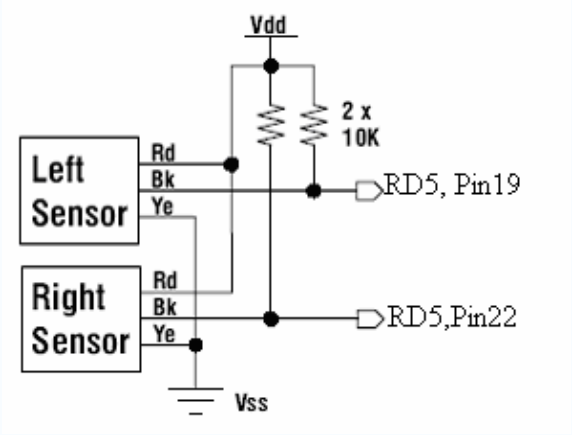

Figure 2.45 : Encoder wiring diagram

2.3.8 Radio Frequency Communication Module Connection

Table 2.2 shows the pin description of the LPRS ER400TRS wireless communication module illustrated in Figure 2.46. The serial inputs and outputs are used for connection to the UART of the PIC18F452 microcontroller. The 'Host Ready Input' is tied to 0 Volt (Ground). Pins 8 and 9 are used for supplying power to the module. Antenna is connected to the Pin 1. Pin 3 is not used.

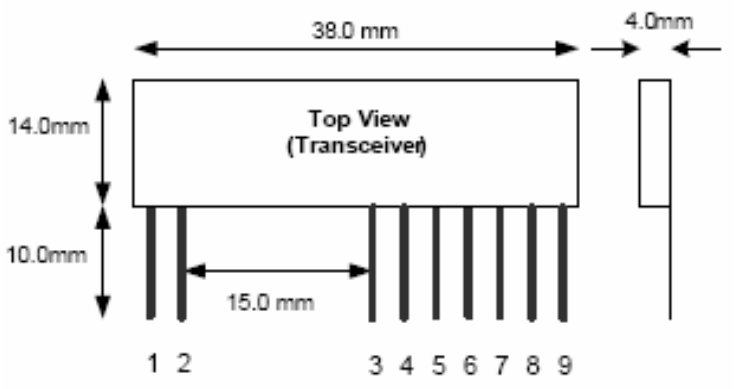

Figure 2.46 : ER400TRS pin diagram

Table 2.2 : LPRS ER400TRS Pin Description

Pin No	Name	Description
1	Antenna	50Ω RF input/output. Connect to suitable antenna.
2	RF Ground	RF ground. Connect to antenna ground (coaxial cable
3	RSSI	Received Signal Strength Indication
4	Busy Output	Digital Output to indicate that transceiver is ready to
5	Serial Data Out	Digital output for received data to host
6	Serial Data In	Digital input for serial data to be transmitted
7	Host Ready	Digital Input to indicate that Host is Ready to receive
8	Vcc	Positive supply pin. +2.5 to +5.5 Volts. This should be
9	Ground	Connect to supply 0 Volt and ground plane

Figure 2.47 shows the connections of all the sensors, actuators and microcontroller of the robot. After having completed all these connections, final electronic circuit board of the Robot as depicted in Figure 2.48 is built.

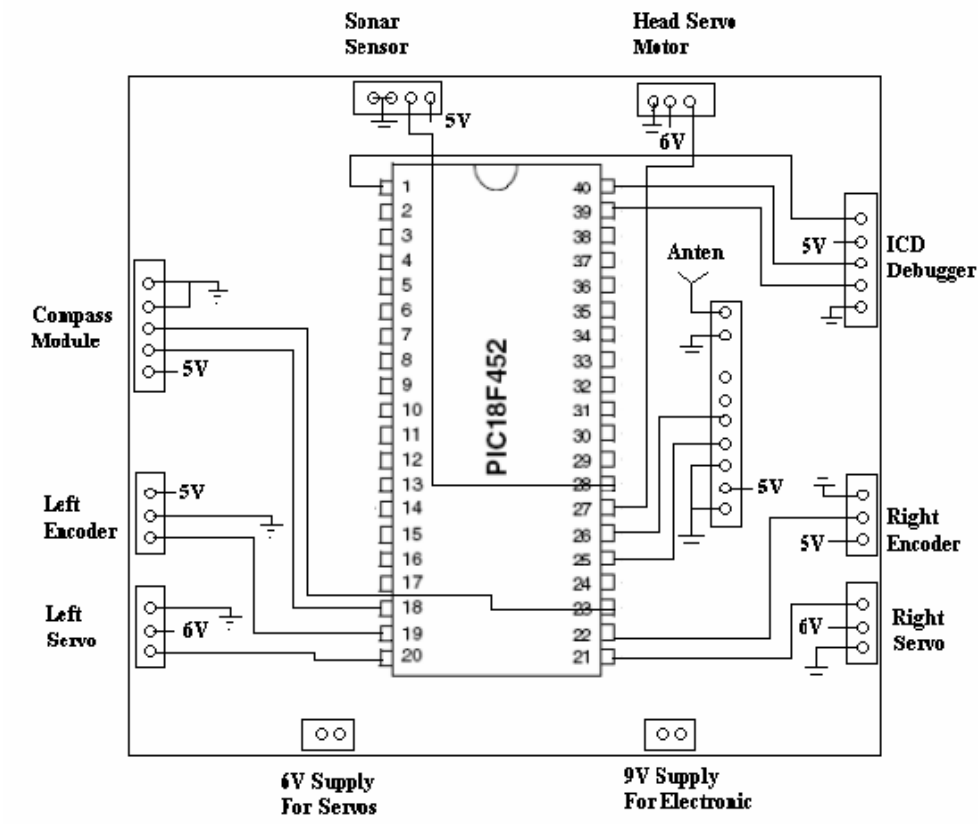

Figure 2.47 : Wiring diagram of the Robot

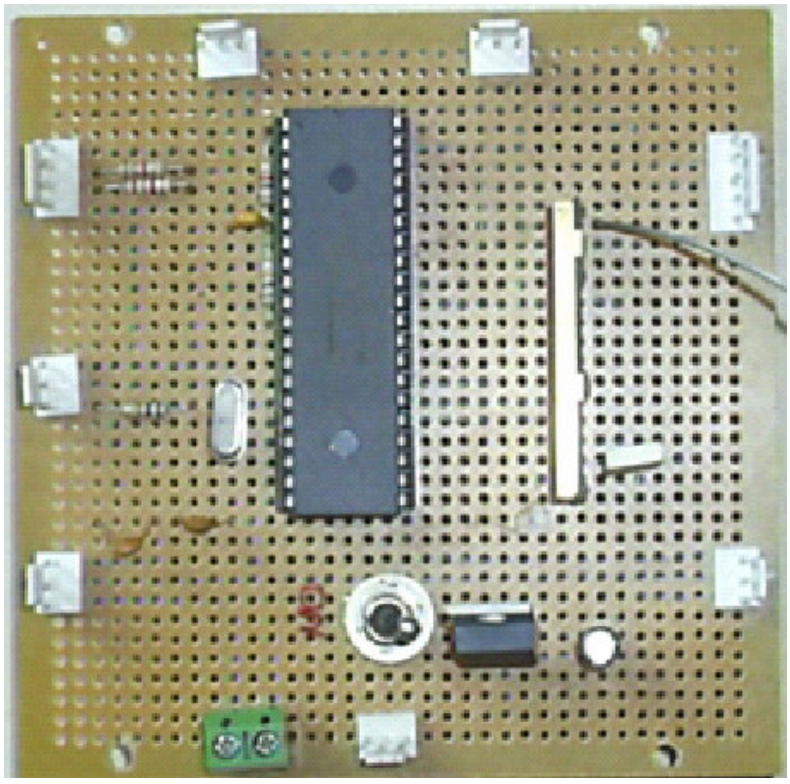

Figure 2.48 : Electronic circuit board of the robot

2.4 Programming the Robot

In the previous chapter, the connections and wirings of the actuators and sensors used by the robot for a number of functions like navigation, object distance measurement have been introduced to you. In this section, the programming used for integrating the different functions as well as creating a high level control program for the robot will be detailed.

The software development tools used in the programming of the robot are Microchip MPLAB IDE and CCS C compiler. MPLAB IDE is a Windows Operating System based Integrated Development Environment for the PIC micro MCU families. CCS C compiler is specifically designed to meet the unique needs of the PIC microcontroller. This allows developers to quickly design applications software in a more readable, high-level language. These tools provide the ability to:

- Create and edit source code using the built-in editor.
- Assemble, compile and link source code.
- Debug the executable logic by watching program flow with the built-in simulator or in real time with in-circuit emulators or in-circuit debuggers.
- Make timing measurements with the simulator or emulator.
- View variables in Watch windows.
- Program firmware into devices with device programmers

2.4.1 Programming the Navigation of the Robot

The robot can be programmed to perform a variety of maneuvers: forward, backward, rotate left, rotate right, and pivoting turns. Figure 2.49 shows the Robot's front, back, left, and right sides.

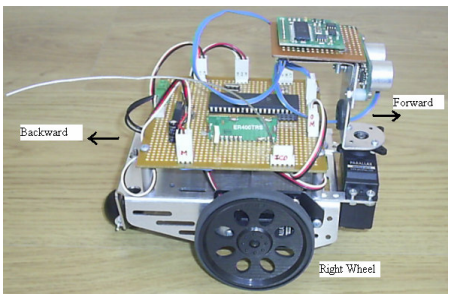

Figure 2.49: The Robot's orientation

Programming the robot to travel a pre-defined distance can be accomplished by two methods. One is measuring the distance it travels in one second, with the help of a ruler and determine the velocity of the robot. Using this velocity, the run time required to cover a desired distance can be calculated. The other way used also in this project is getting feedback from the optical encoder and converting this data to the distance it moved and control the movement depending on the target distance.

The robot also can be started or stopped with ramping. Ramping is a way to gradually increase or decrease the speed of the servos instead of abruptly starting or stopping. This technique can increase the life expectancy of both robot's batteries and servos.

The servo motor of the wheels is controlled by pulsing of its control signal line. Pulse width controls speed and direction. Controlling a servo motor's speed and direction involves a program that makes the PIC18F452 send the same message, over and over again. The message has to repeat itself around 50 times per second for the servo to maintain its speed and direction. Figure 2.50 shows how a Parallax Continuous Rotation servo turns full speed clockwise when it is sent 1.3 ms pulses.

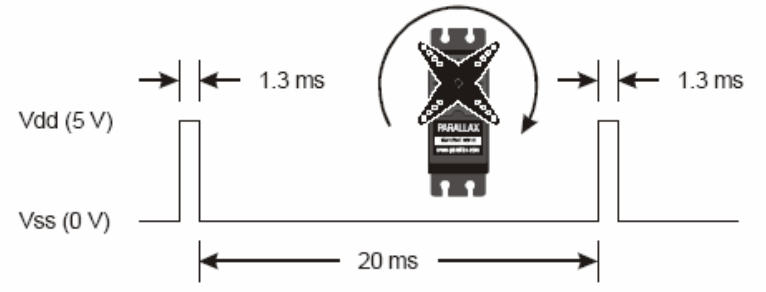

Figure 2.50 : Full speed clockwise

Figure 2.51 shows 1.7 ms pulses sent to the servo. This will make the servo turn full speed counterclockwise.

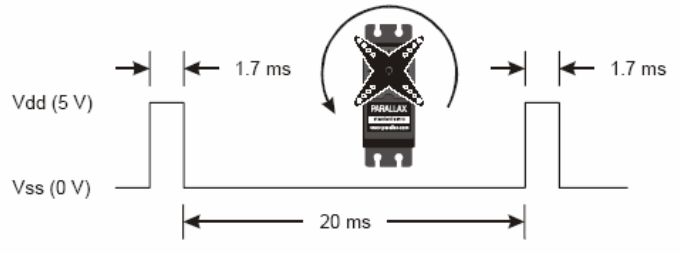

Figure 2.51 : Full speed counterclockwise

Moving Forward : The Robot's left wheel has to turn counterclockwise, but its right wheel has to turn clockwise to make the Robot go forward. Therefore PIC18F452 has to send pulse width between 1.5-1.7ms to the left servo while it has to send pulse width between 1.3-1.5ms to the right servo depending on the speed needed.

Moving Backward : The Robot's left wheel has to turn clockwise, but its right wheel has to turn counterclockwise to make the Robot go backward. Therefore PIC18F452 has to send pulse width between 1.3-1.5ms to the left servo while it has to send pulse width between 1.5-1.7ms to the right servo depending on the speed needed.

Turning Left: The Robot's both wheels have to rotate clockwise to make the Robot turn left. Pulse width between 1.3-1.5ms determines turning speed.

Turning Right : The Robot's both wheels have to rotate counterclockwise to make the Robot turn left. Pulse width between 1.5-1.7ms determines turning speed.

Pivoting: Robot can also be made to turn by pivoting around one wheel. The trick is to keep one wheel still while the other rotates. For example, if you keep the left wheel still and make the right wheel turn clockwise (forward), the robot will pivot to the left. Table 2.3 summarizes the pulse width versus speed and direction relationship.

Table 2.3 : Pulse width versus speed and direction relationship

RD1 Pin 20	RD2 Pin21	Description	Behavior
1.7ms	1.3ms	Both Full Speed	Forward
1.3ms	1.7ms	Both Full Speed	Backward
1.7ms	1.7ms	Both Full Speed	Right turn
1.3ms	1.3ms	Both Full Speed	Left Turn
1.5ms	1.7ms	Left Stop, Right Full Speed	Pivot Back Left
1.3ms	1.5ms	Right Stop, Left Full Speed	Pivot Back Right
1.5ms	1.5ms	Both Stop	Stopped
1.55ms	1.45ms	Both Slow	Forward Slow
1.6ms	1.4ms	Both Medium	Forward Medium
1.7ms	1.4ms	Left Full Speed Right Medium	Veer Right

In summary, Controlling a servo motor's speed and direction involves pulses whose width changes between 1.3ms and 1.7ms depending on the requirement of the speed and direction. In the following source code developed by using the CCS C computer language, the movements of the motor can be controlled by modifying the values of the variables "NumOfCountForLeftServo" and "NumberOfCountForRightServo" respectively.

CCS C source code for the movements of the robot can be found in Appendix A.

2.4.1.1 Calibration of the Servos

Because the servos are not pre-adjusted at the factory, they need to be calibrated. By means of a screwdriver, servo is adjusted so that they stay still. This is called centering the servos. Figure 2.52 shows the signal that has to be sent to the servo to calibrate it. This is called the center signal, and after the servo has been properly adjusted, this signal instructs it to stay still. The instruction consists of a series of 1.5 ms pulses with 20 ms pauses between each pulse.

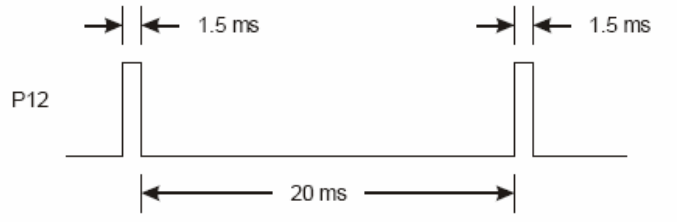

Figure 2.52 : Timing diagram for standstill

When these pulses are generated and the servo is first powered and connected, its horn will start turning, and you will be able to hear the motor inside making a whining noise. By using a screwdriver, the potentiometer in the servo is gently adjusted until finding the setting that makes the servo stop turning as shown in Figure 2.53.

Figure 2.53 : Center adjusting a servo

2.4.2 Object Distance Measurement

The SRF05 Timing diagrams are shown in Figure 2.54. It is only needed to supply a short 10uS pulse to the trigger input to start the ranging. The SRF05 will send out an 8 cycle burst of ultrasound at 40khz and raise its echo line high. It then listens for an echo, and as soon as it detects one it lowers the echo line again. The echo line is therefore a pulse whose width is proportional to the distance to the object. By timing the pulse it is possible to calculate the range in inches/centimeters or anything else. If nothing is detected then the SRF05 will lower its echo line anyway after about 30mS.

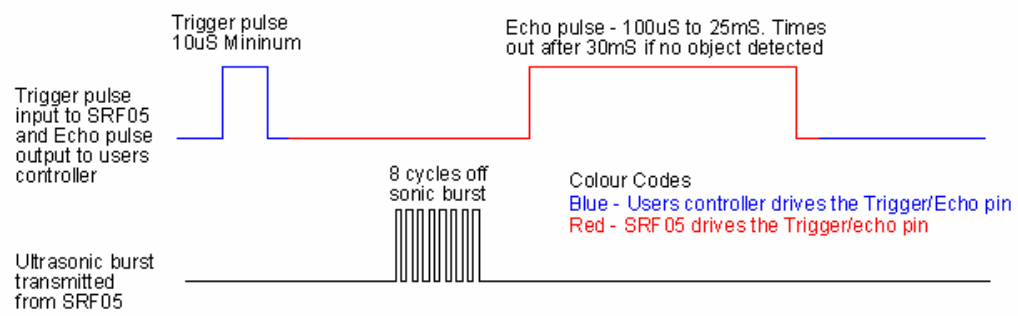

Figure 2.54 : SRF05 timing diagram

The SRF05 provides an echo pulse proportional to distance. If the width of the pulse is measured in µS, then dividing by 58 will give you the distance in cm, uS/58=cm. The SRF05 can be triggered as fast as every 50mS, or 20 times each second. We should wait 50ms before the next trigger, even if the SRF05 detects a close object and the echo pulse is shorter. This is to ensure the ultrasonic "beep" has faded away and will not cause a false echo on the next ranging.

CCS C source code for trigger/echo pulse timing and object distance measurement can be found in Appendix B.

2.4.2.1 Accuracy of Sonar Range Finder

To determine the accuracy of sonar range finder, a number of measurements are done with this sensor . A 20 × 20 × 20 cm cubic object is placed in front of the robot with 25cm intervals, that is the object is first located 25cm away from the robot, then 50, 75, 100, 125 and at last 150cm respectively. Twenty measurements are taken at each distance. The resolution of the measurement system is 1cm. Appendix C shows the results of these measurements. When the distance of the object to the

robot exceeds 50cm, errors start to increase up to 2cm as indicated in Figure 2.55. Many factors may affect this accuracy. The speed of sound in air is approx. 346m/S at 24 degrees C. At 40KHz the wavelength is 8.65mm. The sonar's detect the echo by listening for the returning wavefronts. This echo has an attack/decay envelope, which means it builds up to a peak then fades away. Depending on which wavefront is the 1st to be strong enough to be detected, which could be the 1st, 2nd or even 3rd, the result can jitter by this much.

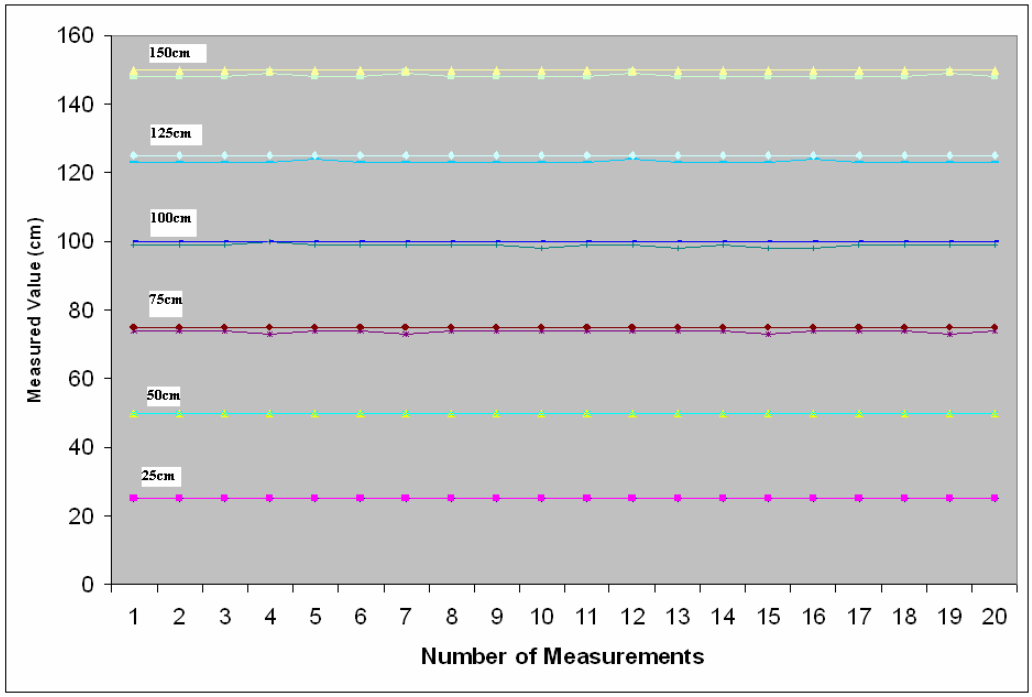

Figure 2.55 : Sonar sensor measurements

The maximum error for a full scale of 1.5m can be calculated by the following formula.

error = ± ((Maximum Error in Range) ÷ (Full Scale Value)) X 100 % **(2.1)**

error = ± (2÷150) X 100 = 1.33 %

2.4.3 Measuring the Orientation by Using CMPS03

I2C interface is used to get the orientation information from the compass module in this project. Figure 2.56 shows the I2C communication protocol. First send a start bit, the module address (0XC0) with the read/write bit low, then the register number.

This is followed by a repeated start and the module address again with the read/write bit high (0XC1). Now, one or two bytes for 8bit or 16bit registers are read respectively. 16bit registers are read high byte first.

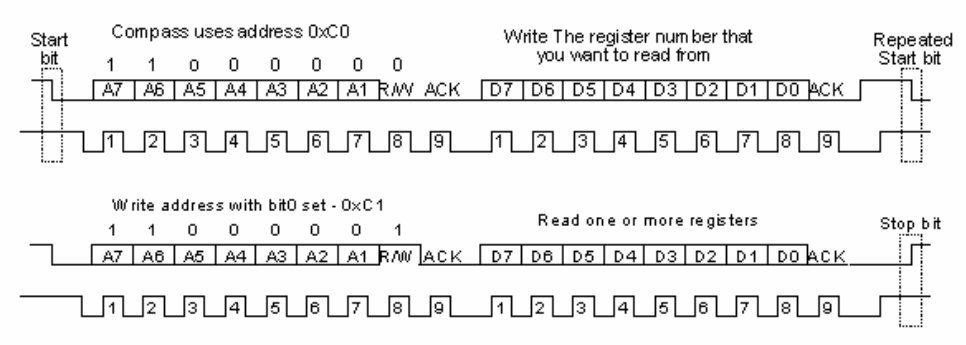

Figure 2.56 : I2C communication protocol

The compass is designed to work at up to the standard clock speed (SCL) of 100KHz. The compass has a 16 byte array of registers, some of which double up as 16 bit registers as illustrated in Table 2.4.

Table 2.4 : Registers of CMPS03 compass module

Register	Function
0	Software Revision Number
1	Compass Bearing as a byte, i.e. 0-255 for a full circle
2,3	Compass Bearing as a word, i.e. 0-3599 for a full circle, representing 0-
4,5	Internal Test - Sensor1 difference signal - 16 bit signed word
6,7	Internal Test - Sensor2 difference signal - 16 bit signed word
8,9	Internal Test - Calibration value 1 - 16 bit signed word
10,11	Internal Test - Calibration value 2 - 16 bit signed word
12	Unused - Read as Zero
13	Unused - Read as Zero
14	Unused - Read as Undefined
15	Calibrate Command - Write 255 to perform calibration step. See text.

CCS C source code for measuring the orientation of the robot can be seen from the Appendix D.

2.4.3.1 Calibration of The Compass Sensor

Calibration only needs to be done once - the calibration data is stored in EEPROM on the PIC16F872 chip. It is not needed to re-calibrate every time the module is powered up. The compass module must be kept flat (horizontal and parallel to the earths surface) with the components on top. Keep the module away from metallic - especially magnetic - objects. To calibrate the module , we only have to write 255 (0xff) to register 15 for each of the four major compass points North, East, South and West. The 255 is cleared internally automatically after each point is calibrated. The compass points can be set in any order, but all four points must be calibrated. For example:

- Set the compass module flat, pointing north. Write 255 to register 15
- Set the compass module flat, pointing east. Write 255 to register 15
- Set the compass module flat, pointing south. Write 255 to register 15
- Set the compass module flat, pointing west. Write 255 to register 15

2.4.4 Measuring How Far the Robot Moves

As mentioned before, each wheel on the robot has an encoder that maintains a counter indicating the angular position of the wheel. As the wheel rotates forward, the counter on the PIC18F452 microcontroller counts up. The distance the wheel travels across the floor with each count can be calculated by dividing the circumference of the wheel by the number of counts the encoder increments with one full revolution of the wheel. At the end, we find the total distance traveled by the Robot by multiplying the total number of counts and the distance per count when the robot stops.

CCS C source code for measuring the distance the robot travels can be found in Appendix E.

3. LOCALIZATION AND MAPPING THEORY

Although the robots we see in science fiction movies appear to navigate with effortless precision, in reality mobile robot navigation is a difficult research problem. The construction of completely autonomous mobile robots is a fundamental objective in the research of robotics. An autonomous mobile robot should be able to determine where it is, relative to a map of the environment to navigate from one place to another. In many situations it is infeasible to provide a priori map of the environment. This might be because mapping the environment by hands is too costly or because the environments changes too rapidly. Because a great deal of navigation tasks is based on an environment map, map building of the environment is important and essential for autonomous mobile robots. Therefore, mobile robot localization and mapping, the process of simultaneously tracking the position of a mobile robot relative to its environment and building a map of the environment, has been a central research topic in mobile robotics. Accurate localization is a prerequisite for building a good map, and having an accurate map is essential for good localization.

3.1 Historical Overview

Robotic mapping research has a long history. In the 1980s and early 1990s, the field of mapping was widely divided into metric and topological approaches. Metric maps capture the geometric properties of the environment, whereas topological maps describe the connectivity of different places [7]. An early representative of the former approach was Elfes and Moravec's important occupancy grid mapping algorithm [8], which represents maps by fine-grained grids that model the occupied and free space of the environment. This approach has been used in a great number of robotic systems, such as [9,10,11]. An alternative metric mapping algorithm was proposed by Chatila and Laumond [12], using sets of polyhedra to describe the geometry of environments. Topological maps represent environments as a list of significant places that are connected via arcs.

Since the 1990s, the field of robot mapping has been dominated by probabilistic techniques. A series of seminal papers by Smith, Self, and Cheeseman [13,14] introduced a powerful statistical framework for simultaneously solving the mapping problem and the induced problem of localizing the robot relative to its growing map. Since then, robotic mapping has commonly been referred to as SLAM or CML, which is short for simultaneous localization and mapping [15], and concurrent mapping and localization [16], respectively. One family of probabilistic approaches employ Kalman filters to estimate the map and the robot location [17,18]. The resulting maps usually describe the location of landmarks, or significant features in the environment. An alternative family of algorithms is based on Dempster's expectation maximization algorithm [19]. These approaches specifically address the correspondence problem in mapping, which is the problem of determining whether sensor measurement recorded at different points in time correspond to the same physical entity in the real world [7].

3.2 Uncertainty in Robotic Mapping

Robots have to be able to accommodate the enormous uncertainty that exists in the physical world. There are a number of factors that contribute to a robot's uncertainty.

Sensors are limited in what they can perceive. Limitations arise from several factors. The range and resolution of a sensor is subject to physical limitations. Sensors are also subject to noise, which perturbs sensor measurements in unpredictable ways and hence limits the information that can be extracted. And finally, sensors can break, detecting a faulty sensor can be extremely difficult [20].

Robot actuation involves motors that are, at least to some extent, unpredictable. Uncertainty arises from affects like control noise, wear-and-tear, and mechanical failure. Some actuators, such as heavy duty industrial robot arms, are quite accurate and reliable. Others, like low cost mobile robots, can be extremely [20].

A key challenge in robotic mapping arises from the nature of the measurement noise. Modeling problems, such as robotic mapping, are usually relatively easy to solve if the noise in different measurements is statistically independent. If this were the case, a robot could simply take more and more measurements to cancel out the effects of

the noise. Unfortunately, in robotic mapping, the measurement errors are statistically dependent. This is because errors in control accumulate over time, and they affect the way future sensor measurements are interpreted. This is illustrated in Figure 3.1, which shows an example path of a mobile robot in a given map of the environment.

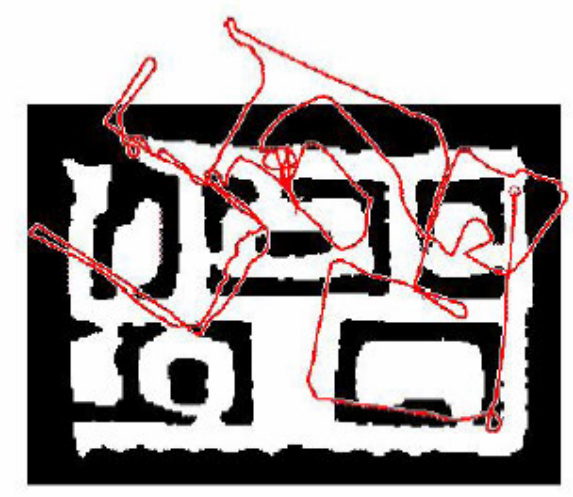

Figure 3.1: Example of accumulated odometry error [7]

As this example shows, a small rotational error on one end of a long corridor can lead to many meters of error on the other. As a result, whatever a robot infers about its environment is plagued by systematic, correlated errors. Accommodating such systematic errors is key to building maps successfully, and it is also a key complicating factor in robotic mapping [7].

Virtually all state-of-the-art algorithms for robotic mapping in the literature have one common feature: They are probabilistic. They all employ probabilistic models of the robot and its environment, and they all rely on probabilistic inference for turning sensor measurements into maps. The reason for the popularity of probabilistic techniques stems from the fact that robot mapping is characterized by above uncertainties and sensor noise. Probabilistic algorithms approach the problem by explicitly modeling different sources of noise and their effects on the measurements. In the evolution of mapping algorithms, probabilistic algorithms have emerged as the sole winner for this difficult problem [7].

3.2.1 Bayes Rule

The basic principle underlying virtually every single successful mapping algorithm is Bayes rule:

$$p(x \mid d) = \eta \, p(d \mid x) \, p(x) \qquad \textbf{(3.1)}$$

Bayes rule is the archetype of probabilistic inference. Suppose we want to learn about a quantity x (e.g., a map), based on measurement data d (e.g., range scans, odometry). Then Bayes rule tells us that the problem can be solved by multiplying two terms: $p(d \mid x)$ and $p(x)$. The term $p(d \mid x)$ specifies the probability of observing the measurement d under the hypothesis x. Thus, $p(d \mid x)$ is a generative model, in that it describes the process of generating sensor measurements under different worlds x. The term $p(x)$ is called the prior. It specifies our willingness to assume that x is the case in the world before the arrival of any data. Finally, η is a normalizer that is necessary to ensure that the left hand side of Bayes rule is indeed a valid probability distribution [7].

3.2.2 Kalman Filter

One common approach to solving the SLAM problem is to use a Kalman filter to simultaneously estimate the position of a moving vehicle along with the positions of landmarks seen by the vehicle. The Kalman Filter is one of the most useful estimation tools available today. Loosely speaking, Kalman filtering provides a recursive method of estimating the state of a dynamical system in the presence of noise [21,22]. Kalman filtering is a form of probabilistic estimation where the estimate is assumed to be a Gaussian PDF. The probabilistic representation allows for the uncertainties that arise from uncertain motion models and noisy sensor readings to be accounted for in a principled way. The challegenge is then to maintain a position probability density over all possible robot poses. The extended Kalman filter (EKF), which is a Kalman filtering variant, can be used on nonlinear systems [23].

3.3 Localization

Mobile robot localization is the problem of determining the pose of a robot relative to a given map of the environment. It is often called position estimation. The robot is given a map of its environment and its goal is to determine its position relative to this map given the perceptions of the environment and its movements [20].

In summary, localization is the process of determining the robot's location within the environment. More precisely, it is a procedure that takes as input

(i) a map
(ii) an estimate of the robot's current pose
(iii) a set of sensor readings

and produces as output a new estimate of the robot's current pose. Any of (i), (ii), and (iii) may be incomplete, and all are toleranced by error bounds. For example, the map may lack information, the pose estimate may contain only orientation information, and the set of sensor readings may have elements with impossible values [24].

This section introduces you a number of methods for mobile robot localization.

3.3.1 Markov Localization

Probabilistic localization algorithms are variants of the Bayes filter . The straightforward application of Bayes filter to the localization problem is called Markov Localization. Markov localization addresses the global localization problem, the position tracking problem, and the kidnapped robot problem in static environments.

3.3.2 Grid Localization

Grid localization approximates the posterior using a histogram filter over a grid decomposition of the pose space. Grid localization requires as input the discrete probability values $\{p_{k,t}\}$ along with the most recent measurement, control and the map. The discrete Bayes filter maintains as posterior a collection of discrete probability values

$$bel(x_t)=\{p_{k,t}\} \quad \textbf{(3.2)}$$

Where each probability $p_{k,t}$ is defined over a grid cell x_k.The set of all grid cells forms a partition of the space of all legitimate poses:

$$Domain(X_t) = x_{1,t} \cup x_{2,t} \cup ....\cup x_{k,t} \quad \textbf{(3.3)}$$

A number of issues arise when implementing grid localization: with a fine grained grid, the computation required for naive implementation may make the algorithm intolerably slow. With a coarse grid, the additional information loss through the dicretization negatively affects the filter and if properly treated may even prevent the filter from working [20].

3.3.3 Monte Carlo Localization

Monte Carlo Localization(MCL) algorithm is arguably the most popular localization algorithm to date. Despite its relatively young age, MCL has already become one of the most popular localization algorithms in robotics. It is easy to implement and tends to work well across a broad range of localization problems. It uses particle filters to estimate posteriors over robot poses. Like Grid based localization, MCL is applicable to both local and global localization problems [20].

3.3.4 Odometry

Odometry is the study of position estimation during wheeled vehicle navigation. The term is also sometimes used to describe the distance traveled by a wheeled vehicle. Odometry is used by some track or wheeled robots to estimate (not determine) their position relative to a starting location. Odometry is the use of data from the rotation of wheels or tracks to estimate change in position over time. This method is often very sensitive to error. Rapid and accurate data collection, equipment calibration, and processing are required in most cases for odometry to be used effectively.

The word odometry is composed from the Greek words hodos (meaning "travel", "journey") and metron (meaning "measure") [25].

3.3.5 Dead Reckoning

Dead reckoning (DR) is the process of estimating one's current position based upon a previously determined position, and advancing that position based upon known speed, elapsed time, and course. While this method of navigation is no longer considered primary in aircraft, automobiles, rail engines, and construction site engines (tunnels), it is still often used as a backup in case of failure of the electronic navigation systems. Dead Reckoning has recently found new life in mobile robotics. Although fast and relatively easy to implement, dead reckoning suffers from accumulation of errors. Dead reckoning is only useful if a position correction can somehow be given, so that any errors accumulated since the last correction are zeroed out.

3.4 Robotic Mapping

The problem of robotic mapping is that of acquiring a spatial model of a robot's environment. Maps are commonly used for robot navigation (e.g., localization) [26, 27]. To acquire a map, robots must possess sensors that enable it to perceive the outside world. Sensors commonly brought to bear for this task include cameras, range finders using sonar, laser, and infrared technology, radar, tactile sensors, compasses, and GPS. Most robot sensors are subject to strict range limitations. For example, light and sound cannot penetrate walls. These range limitations makes it necessary for a robot to navigate through its environment when building a map. The motion commands (controls) issued during environment exploration carry important information for building maps, since they convey information about the locations at which different sensor measurements were taken [7].

3.4.1 Kalman Filter Approach

A classical approach to generating maps is based on Kalman filters [21]. This approach can be traced back to a highly influential series of papers by Smith, Self, and Cheeseman [13], who in 1985 through 1990 proposed a mathematical formulation of the approach that is still in widespread use today. In the following years, a number of researchers developed this approach further [28], most notably a group of researchers located at the University of Sydney in Australia. In the

literature, Kalman filter-based mapping algorithms are often referred to as SLAM algorithms, where SLAM stands for simultaneous localization and mapping.

3.4.2 Expectation Maximization Algorithms

A recent alternative to the Kalman filter paradigm is known as the expectation maximization family of algorithms, or in short, EM. EM is a statistical algorithm that was developed in the context of maximum likelihood (ML) estimation with latent variables, in a influential paper by Dempster, Laird and Rubin [19]. A recent book on this topic illustrates the wealth of research that presently exists on the EM algorithm [29].

3.4.3 Occupancy Grid Maps

The mapping algorithms described above all address the mapping problem with unknown robot poses which, as pointed out above, is known as the simultaneous localization and mapping (SLAM) problem. The simpler case—mapping with known poses—has also received attention in the literature. One mapping algorithm, known as occupancy grid maps developed by Elfes and Moravec in the mid-Eighties [8], has enjoyed enormous popularity.

The central problem addressed by occupancy grid mapping and related algorithm is the problem of generating a consistent metric map from noisy or incomplete sensor data. Even if the robot poses are known, it is sometimes difficult to say whether a place in the environment is occupied or not, due to ambiguities in the sensor data. The best-explored applications of occupancy grid maps require robots with range sensors, such as sonar sensors or laser range finders. Both sensors are characterized by noise. Sonars, in addition, cover an entire cone in space, and from a single sonar measurement it is impossible to say where in the cone the object is. Both sensors are also sensitive to the angle of an object surface relative to the sensor and the reflective properties of the surface (absorption and dispersion) [7].

A 2-dimensional grid is used to provide a map of the robot's environment. The grid map consists of a matrix of cells, each containing an occupancy value and a certainty value. These values are used by the occupancy and localization algorithms respectively.

The occupancy algorithm creates a map of the environment by integrating data collected over time. As the robot explores its environment, information from range sensor sweeps is combined with information about the robot's location to update the occupancy values for the global grid map. Thus the occupancy value for each cell in the grid indicates whether the cell is occupied, empty or unexplored. Due to the inaccuracy of sonar range sensors it is not sufficient to simply mark the cell at the end of the range vector as occupied. Instead we need to be able to specify an occupancy probability and integrate range readings over time. Bayes' rule is used to estimate this probability. The occupancy value ranges from 0 to 1. An initial value of 0.5 is used to mark this cell as undecided (unexplored). Then thresholds can be used to determine if the cell is occupied, empty or unexplored [30].

4. LOCALIZATION AND MAPPING ALGORITHM

There are many technologies available for robot localization and mapping as explained above. In each approach, however, improvements in accuracy come at the cost of expensive hardware and additional processing power. In this study, odometry and occupancy-grid methods are used for localization and mapping applications respectively.

4.1 Localization Algorithm

Location estimation using odometry information is a very common approach. Each wheel on the robot has an encoder that maintains a counter indicating the angular position of the wheel. As the wheel rotates forward, the counter counts up. The distance the wheel travels across the floor with each count can be calculated by dividing the circumference of the wheel by the number of counts the encoder increments with one full revolution of the wheel.

By closely monitoring the movement of its wheels using encoders, the robot can track its movements. Using basic geometry it can calculate and deduce changes in its position caused by each small movement.

The diameter of the wheels of the Robot is 67mm. As each wheel turns, an encoder outputs 8 equally-spaced pulses per revolution. Each of these pulses will correspond to a distance of travel D equal to the wheel circumference divided by n, or

Distance per Count :

$$D = pi \times d / n = 26mm \quad \textbf{(4.1)}$$

where pi= 3.14159265

d= 67.06 mm (diameter of the wheel)

n= 8 (number of pulses per revolution)

each pulse corresponds to 26 mm as illustrated in Figure 4.1.

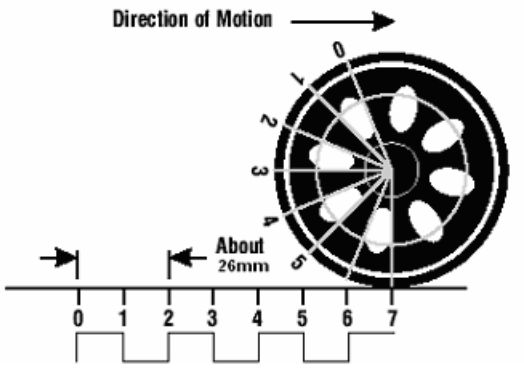

Figure 4.1 : Encoder output

When the robot moves in approximately a straight line, the distance it travels is simply the number of encoder counts times the distance per encoder count:

DeltaDistance = (Encoder Counts) * (Distance per Count) **(4.2)**

The localization algorithm can determine the robot's position along an arbitrary path by treating its motion as many small, discrete movements. By adding all of the discrete movements together, the algorithm can deduce the robot's position regardless of the path it follows.

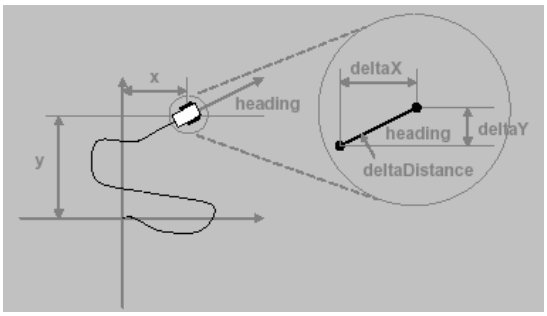

Figure 4.2 : Localization along an arbitrary path

As shown in above figure, imagine the robot moves a small distance, deltaDistance, while it travels forward in the heading direction measured by the digital compass module. Assuming the direction the robot is heading doesn't change significantly during each small movement, the change in the position along the x axis, deltaX, and

change in position along the y axis, deltaY, can be calculated using the following trigonometric calculations:

deltaX = deltaDistance * cos(heading) **(4.3)**

deltaY = deltaDistance * sin(heading) **(4.4)**

The deltaX and deltaY values can then be added to the previous position estimate to obtain a new position estimate.

4.2 Mapping Algorithm

Occupancy grid method is used for mapping in this project. A 2-dimensional grid is used to provide a map of the robot's environment. The grid map consists of a matrix of cells, each containing an occupancy value. This value is used by the occupancy grid algorithm. The size of the map used in this study was 10x10 cells and the cell size was 20cm. The test environment can be regarded as a simple room of the 2m x 2m dimensions. A typical move-sweep-localize-update cycle was used to construct the map as shown in Figure 4.3. First the robot is moved to a new location and a 180 degrees range sensor sweep is made. Using the odometry data localization is performed and finally using the newly found location, the global map is updated with the range data. The cycle repeats until the complete floor plan is explored.

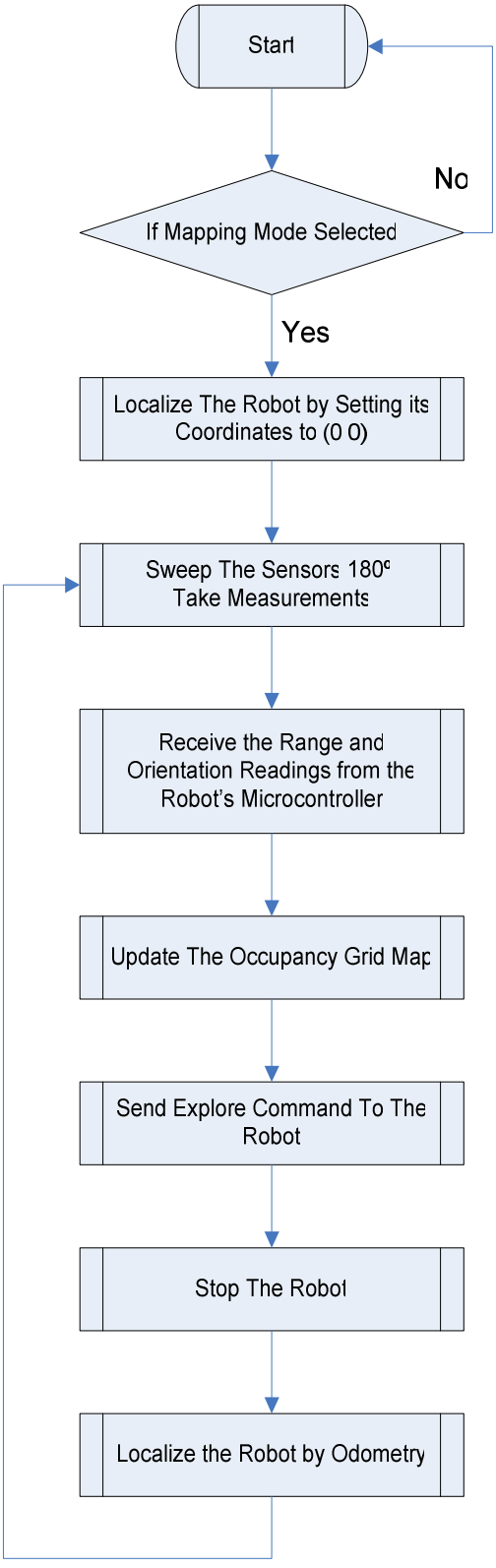

Figure 4.3 : Localization and mapping block diagram

4.3 Obstacle Avoidance Algorithm

The robot is able to detect obstacles that it encountered while moving and change its path in order not to collide with them. The ultrasonic range finder is used to measure the proximity of the obstacles in the area continuously. Figure 4.4 depicts the simple flow charts of the algorithm. Here 25 cm is selected arbitrarily.

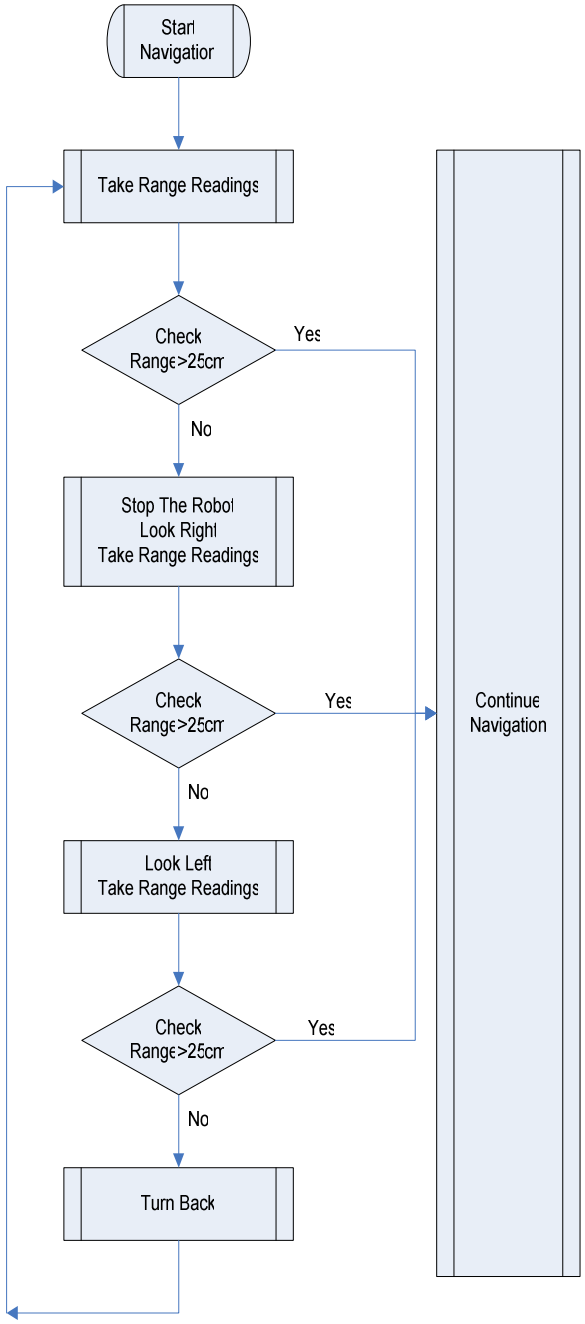

Figure 4.4 : Obstacle avoidance block diagram

4.4. The Application Software

In this project, C Sharp 2.0 is used as the development software to be able to control the robot from the remote workstation and perform mapping and localization. The development environment is Visual Studio 2005. Figure 4.5 shows the main screen of the application software.

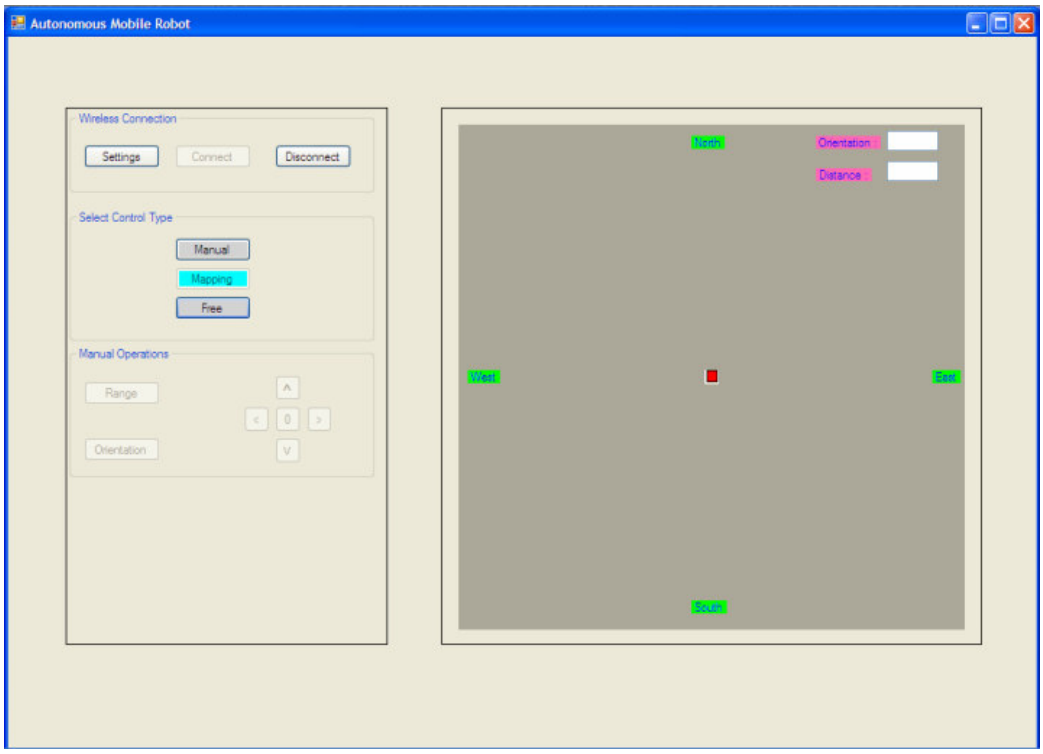

Figure 4.5 : Main control page of the application software

4.4.1 Com Port Settings

As explained before, the communication between the robot and computer is realized through the wireless radio frequency link by using RS232 serial communication protocol. The software always listens the specified serial port and as soon as new data arrives it calls a dedicated procedure for acquiring the new data having sent from the robot. The com port settings of the computer must be edited in order for healty communication by using the specified screen as shown in Figure 4.6.

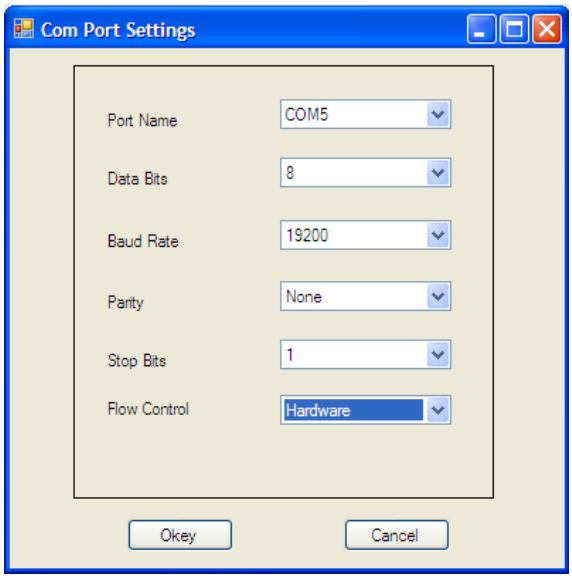

Figure 4.6 : Com port settings

4.4.2 Operation Modes of the Robot

The robot has three modes of operation. Figure 4.7 shows these operation modes of the robot. Once the com port settings are done and communication is established, the robot is ready to be controlled from the remote computer.

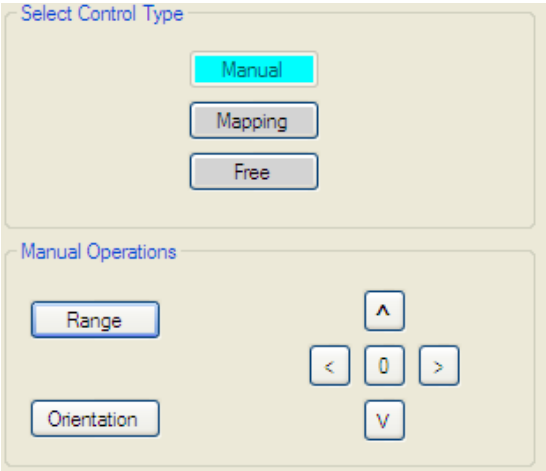

Figure 4.7 : Operation modes of the robot

In “Manual” control mode, the robot waits for user commands. In this mode, the control is in the user and the user can control the navigation of the robot such as forward, backward, turn left and right by using dedicated buttons. Moreover the user

is able to measure the range and orientation of the robot by using related buttons on the screen as shown in Figure 4.8 and 4.9.

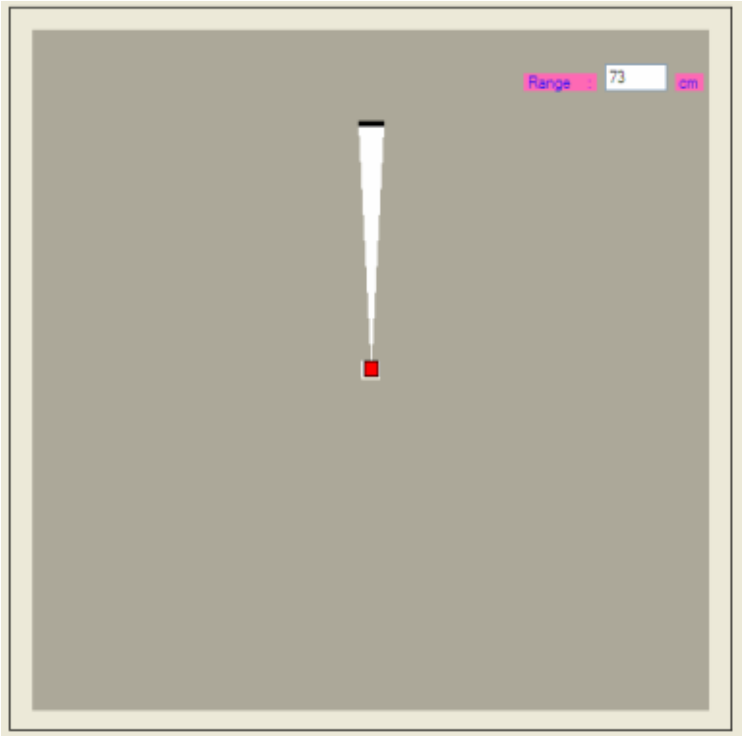

Figure 4.8 : Measuring the range manually

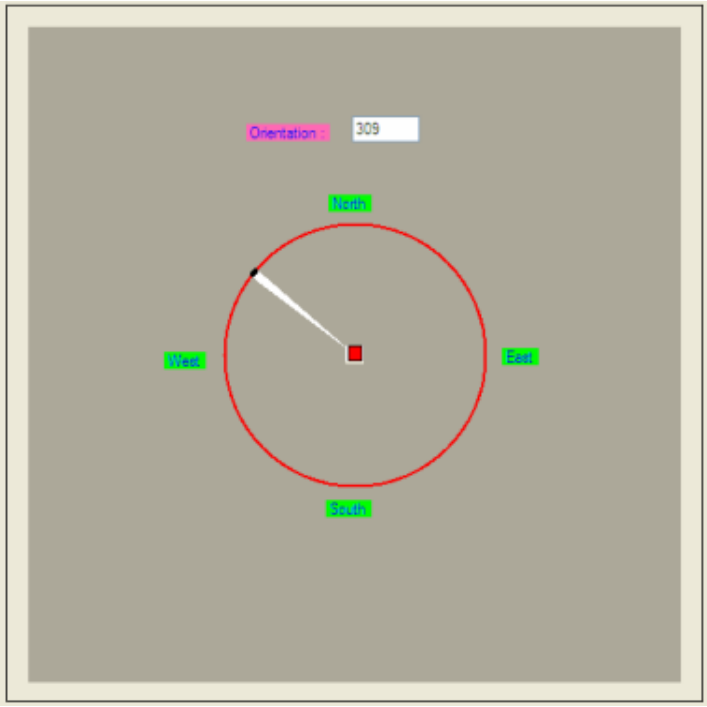

Figure 4.9 : Measuring the orientation manually

In the “Free” mode, the robot navigates through the environment freely. In the “Mapping” mode, the robot performs mapping and localization.

4.5 Accuracy of Localization Algorithm

As described earlier, the robot uses its encoder and compass module to keep track of its position. A simple experiment is performed in order to determine how closely the algorithm approximates the robot’s correct position after it moves 103 cm along with an arbitrary direction. Figure 4.10 shows the new position of the robot after 103 cm of navigation in the direction of 320º. The calculated displacement using number of encoder pulses is obtained as 101.35 cm between first and last position of the robot, when the actual displacement is 103 cm. Table 4.1 shows actual and measured displacements in the West and North directions.

Table 4.1 : Actual and measured displacements

Orientation	Actual(cm)	Measured(cm)	Error (cm)
North	81.16	77.63	3.53
West	63.41	65.14	-1.73
Total	103	101.35	1.65

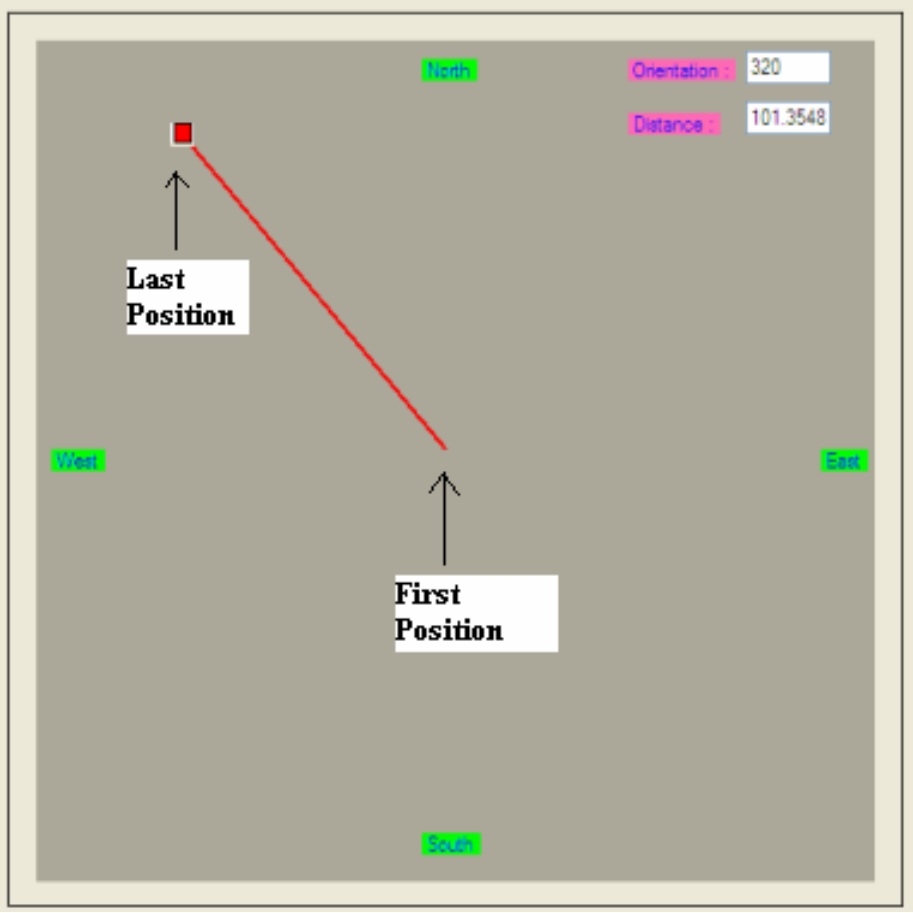

Figure 4.10 : Measured displacement

4.6 Mapping Demonstration

The following screens show how mapping is performed step by step in an empty environment whose size is 1.5 × 1.5m. At first, the robot is located at the point (0,0). After that, it localizes itselft at the point (0,0) and performs first sweep. In Figure 4.11, the map after the first sweep is shown. The robot is represented by a red rectangle. The white regions show the unoccupied fields on the other hand the black regions show occupied cells.

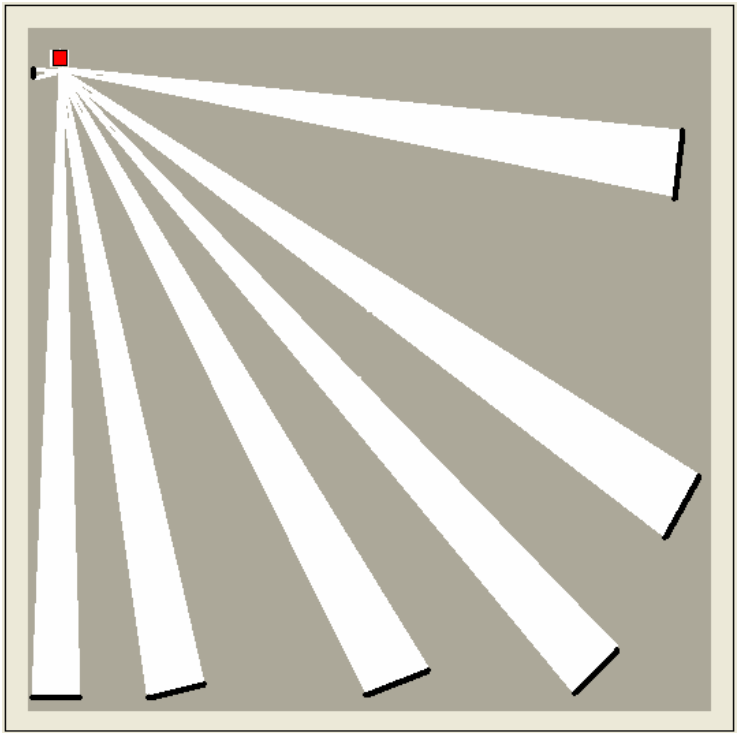

Figure 4.11: First sensor sweep

The robot then moves its next position and localizes itself in this new location. The the second sweep is performed and the map is updated as in the Figure 4.12.

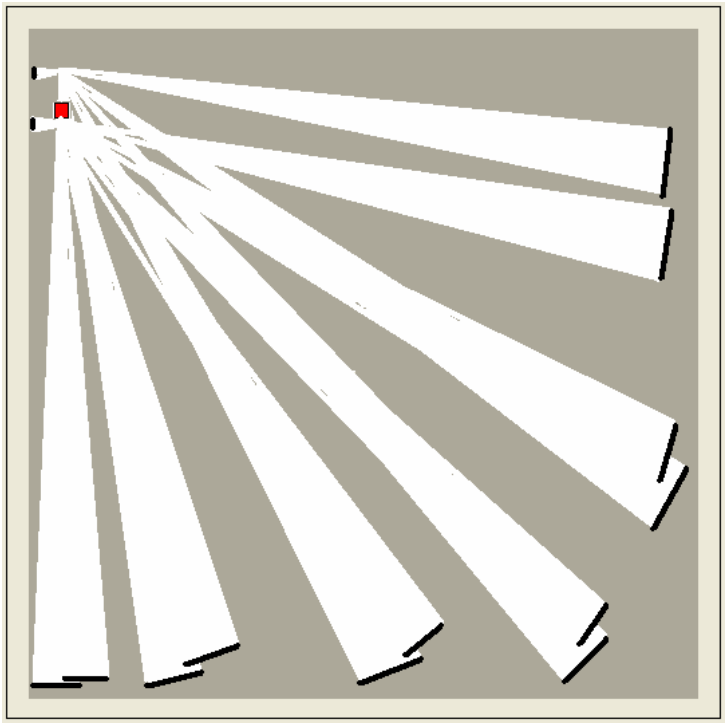

Figure 4.12 : Second sensor sweep

After the third sweep, the the dimension of the test environment is clearer as shown in Figure 4.13.

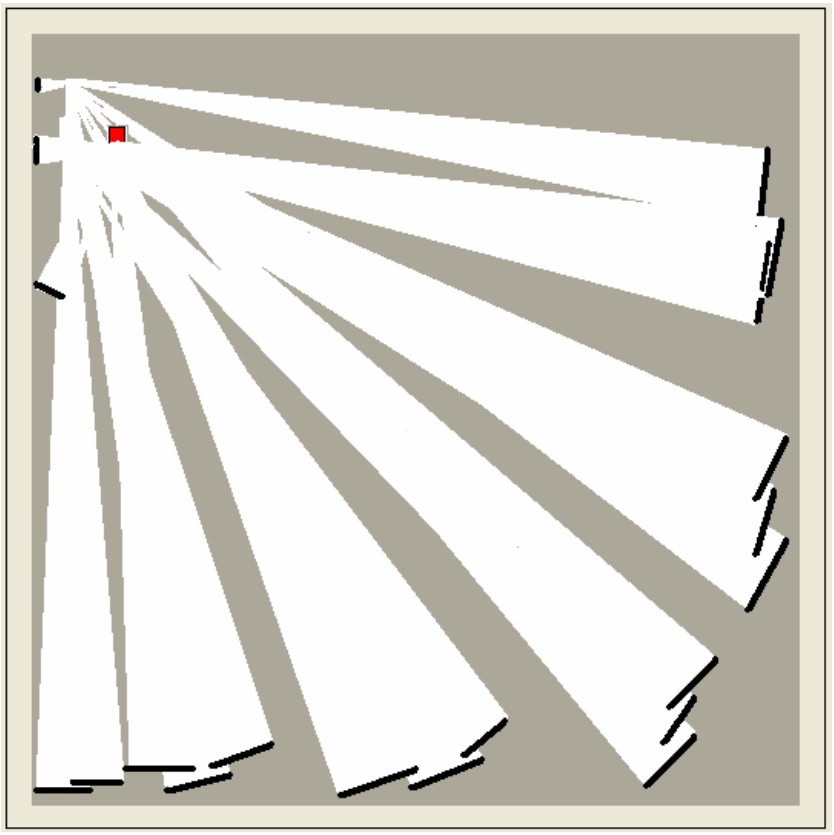

Figure 4.13 : Third sensor sweep

5. RESULTS AND RECOMMENDATIONS

The aim of this project is to create a wheeled mobile robot that can map its environment and locate itself within the environment. This objective is mostly accomplished. The robot was designed and built in such a way that it can be used for localization and mapping applications. The robot was equipped with lots of sensors and actuators that enable it to sense the environment and navigate through the environment without striking the obstacles around it.

The localization accuracy plays a major role in the map construction problem. Odometry is used as a localization method in this project. Although fast and relatively easy to implement, odometry suffers from the accumulation of errors. If an update from two or three bad locations is included in the map, the map will be greatly destroyed. Therefore, major improvements can be made to the localization algorithm used in this project. Probabilistic techniques such as Markov or Monte Carlo Localization as explained in chapter four can be used to improve the localization accuracy. This enables the robot to determine its location from previous measurements. Combined with probabilistic methods, odometry can provide a more accurate estimation of real location by eliminating any accumulated errors.

Probabilistic techniques can also be used to for occupancy grid mapping due to the inaccuracy of sonar range readings. The inaccuracies inherent in a sonar measurement require more than marking occupied or unoccupied cells. Therefore, it will be more useful to specify an occupancy value for each cell in the grid map that indicate whether the cell is occupied, empty or unexplored. Then, the probability that a cell is occupied is calculated and thresholds can be used to determine if the cell is occupied or unoccupied.

REFERENCES

[1] Larry Heath, 1985, Fundamentals of Robotic , Theory and Applications, a Printice Hall Company

[2] The University of Texas, 2006, http://www.robotics.utexas.edu

[3] NASA, 2006, http://prime.jsc.nasa.gov

[4] Panasonic Corporation of North America, 2006, http://www.panasonicfa.com

[5] NASA, 2006, http://www.robotics.jpl.nasa.gov

[6] Parallax, Inc, 2006, http://www.parallax.com

[7] Sebastian Thrun, 2002, Robotic Mapping: A Survey, Carnegie Mellon University, Pittsburgh, CMU-CS-02-111

.[8] A. Elfes, June 1987, Sonar Based Real World Mapping and Navigation. IEEE Journal of Robotics and Automation, RA-3(3):249–265.

[9] J. Borenstein and Y. Koren, June 1991, The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robots. IEEE Journal of Robotics and Automation, 7(3):278–288.

[10] J. Buhmann, W. Burgard, A.B. Cremers, D. Fox, T. Hofmann, F. Schneider, J. Strikos, and S. Thrun, 1995, The Mobile Robot Rhino. AI Magazine, 16(1).

[11] W. Burgard, A.B. Cremers, D. Fox, D. H¨ahnel, G. Lakemeyer, D. Schulz, W. Steiner, 1999, Experiences with an Interactive Museum Tour Guide Robot. Artificial Intelligence, 114(1-2):3–55.

[12] R. Chatila and J.-P. Laumond, 1985, Position Referencing and Consistent World Modeling for Mobile Robots. In Proceedings of the1985 IEEE International Conference on Robotics and Automation.

[13] R. Smith, M. Self, and P. Cheeseman, 1990, Estimating Uncertain Spatial Relationships in Robotics. In I.J. Cox and G.T. Wilfong Editors, Autonomous Robot Vehicles, pages 167–193. Springer-Verlag.

[14] R. C. Smith and P. Cheeseman, 1985, On the Representation and Estimation of Spatial Uncertainty. Technical Report TR 4760 & 7239, SRI.

[15] G. Dissanayake, H. Whyte, and T. Bailey, April 2000, A Computationally Efficient Solution to the Simultaneous Localization and Map Building (SLAM) Problem. Working notes of ICRA 2000 Workshop W4.

[16] S. Thrun, D. Fox, and W. Burgard, 1998, A Probabilistic Approach to Concurrent Mapping and Localization for Mobile Robots. Machine Learning, 31:29–53.

[17] J.A. Castellanos and J.D. Tard´os, 2000, Mobile Robot Localization and Map Building: A Multisensor Fusion Approach. Kluwer Academic Publishers, Boston.

[18] G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba, 2001, A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. IEEE Transactions of Robotics and Automation.

[19] A.P. Dempster, A.N. Laird, and D.B. Rubin, 1977, Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society, Series B, 39(1):1–38.

[20] Dieter Fox, Wolfram Burgard, Sebastian Thrun, 2005 Probabilistic Robotic, The MIT Press.

[21] R.Kalman, 1960, A new Approach to Linear Filtering and Prediction Problems. Trans. of the ASME, Journal of basic engineering, 82:35-45.

[22] P. Maybeck 1990, The Kalman Filter: An Introduction to Concepts. In Autonomous Robot Vehicles, Springer Verlag.

[23] Howie Choset, Kevin M. Lynch, 2005, Principles of Robot Motion, The MIT Press

[24] R. G. Brown and B. R. Donald, 2000, Mobile Robot Self-Localization without Explicit Landmarks, Algorithmica 26: 515–559

[25] Wikimedia Foundation, Inc., 2006. Wikipedia, The Free Encyclopedia http://en.wikipedia.org/wiki/Odometry

[26] J. Borenstein, B. Everett, and L. Feng, 1996, Navigating Mobile Robots: Systems and Techniques. A. K. Peters, Ltd., Wellesley.

[27] D. Kortenkamp, R.P. Bonasso, 1998, AI-based Mobile Robots, Case studies of Successful Robot Systems, Cambridge, MIT Press.

[28] J.A. Castellanos, J.M.M. Montiel, J. Neira, and J.D. Tard´os. The Spmap, 1999, A Probabilistic Framework for Simultaneous Localization and Map Building. IEEE Transactions on Robotics and Automation, 15(5):948–953.

[29] G.J. McLachlan and T. Krishnan, 1997, The EM Algorithm and Extensions, Wiley Series in Probability and Statistics, New York.

[30] Vassilis Varveropoulos, 2006, Robot Localization and Map Construction Using Sonar Data, vassilis@users.sourceforge.net

APPENDIX A

```
#include <18F452.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#define LEFT_SERVO_PIN     PIN_D1
#define RIGHT_SERVO_PIN    PIN_D2
#define TIMER_DIV     2
#define TIMER_RATE      getenv("CLOCK") / 4 / TIMER_DIV
#define Off_TIME     0.0200
#define NumOfCountForOff (int16)((float)TIMER_RATE * Off_TIME)
#define Full_Speed_CW_ON_TIME 0.0014706
#define Full_Speed_CCW_ON_TIME 0.00153
#define Still_On_Time 0.0015
int16 NumOfCountForLeftServo;
int16 NumOfCountForRightServo;
Void Stop_Servos(void)
{
   NumOfCountForLeftServo=(int16)((float)TIMER_RATE * Still_On_Time);
   NumOfCountForRightServo=(int16)((float)TIMER_RATE * Still_On_Time);
   delay_ms(500);
}
void Move_Forward(void)
{
   NumOfCountForLeftServo=(int16)((float)TIMER_RATE                    *
Full_Speed_CCW_ON_TIME);
   NumOfCountForRightServo=(int16)((float)TIMER_RATE                   *
Full_Speed_CW_ON_TIME);
}
void Move_Backward(void)
{
   NumOfCountForLeftServo=(int16)((float)TIMER_RATE                    *
Full_Speed_CW_ON_TIME);
   NumOfCountForRightServo=(int16)((float)TIMER_RATE                   *
Full_Speed_CCW_ON_TIME);
}
void Turn_Right(void)
{
   NumOfCountForLeftServo= (int16)((float)TIMER_RATE * 0.00152);
   NumOfCountForRightServo=(int16)((float)TIMER_RATE * 0.00152);
   delay_ms(2620);
   Stop_Servos();
}
void Turn_Left(void)
{
```

```
   NumOfCountForLeftServo= (int16)((float)TIMER_RATE * 0.00148);
   NumOfCountForRightServo=(int16)((float)TIMER_RATE * 0.00148);
   delay_ms(2620);
   Stop_Servos();
}
void init(void)
{
  setup_ccp1(CCP_COMPARE_INT);
  setup_ccp2(CCP_COMPARE_INT);
  setup_timer_1(T1_DIV_BY_2 | T1_INTERNAL);
  enable_interrupts(INT_CCP1);
  enable_interrupts(INT_CCP2);
  enable_interrupts(GLOBAL);
  Stop_Servos();
}
#int_ccp1
void isr_ccp1()
{
  static int1 TOGGLE_LEFT_Servo = 0;
  if(TOGGLE_LEFT_Servo)
  {
    output_low(LEFT_SERVO_PIN);
    CCP_1 =get_timer1()+NumOfCountForOff-21;
    TOGGLE_LEFT_Servo = 0;
  }
  else
  {
    output_high(LEFT_SERVO_PIN);
    CCP_1 =get_timer1()+NumOfCountForLeftServo-21 ;
    TOGGLE_LEFT_Servo = 1;
  }
}
#int_ccp2
void isr_ccp2()
{
 static int1 TOGGLE_RIGHT_Servo = 0;
 if(TOGGLE_RIGHT_Servo)
  {
    output_low(RIGHT_SERVO_PIN);
    CCP_2 = get_timer1()+NumOfCountForOff-24;
    TOGGLE_RIGHT_Servo = 0;
  }
  else
  {
    output_high(Right_SERVO_PIN);
    CCP_2 =get_timer1()+NumOfCountForRightServo-24;
    TOGGLE_RIGHT_Servo = 1;
  }
}
```

APPENDIX B

```
#include <18F452.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)
#use i2c(Master,sda=PIN_C4,scl=PIN_C3)
#bit PD5Dir = 0xf95.5        // Sonar range data reading pin direction bit(TRISD bit 5)
int16 get_srf05(void)
{
delay_ms(100);
PD5Dir=0;
output_high(PIN_D5);
delay_us(10);
output_low(PIN_D5);
PD5Dir=1;
set_timer1(0);
while ( !input(PIN_D5) );
setup_timer_1 ( T1_INTERNAL | T1_DIV_BY_4 );
while ( input(PIN_D5) );
setup_timer_1 ( T1_DISABLED );
return (int16)(get_timer1()*0.8)/58;  // gives distance in cm, 0.8=(4/20)*4
}
```

APPENDIX C

Table C.1 : Sonar sensor range measurements

Number of Measurement	Object at 25cm	Object at 50cm	Object at 75cm	Object at 100cm	Object at 125cm	Object at 150cm
1	25	50	74	99	123	148
2	25	50	74	99	123	148
3	25	50	74	99	123	148
4	25	50	73	100	123	149
5	25	50	74	100	124	148
6	25	50	74	99	123	148
7	25	50	73	99	123	148
8	25	50	74	99	123	148
9	25	50	74	99	123	148
10	25	50	74	99	123	148
11	25	50	74	99	123	148
12	25	50	74	99	124	149
13	25	50	74	99	123	148
14	25	50	74	99	123	148
15	25	50	73	99	123	148
16	25	50	74	98	124	148
17	25	50	74	99	123	148
18	25	50	74	99	123	148
19	25	50	73	99	123	148
20	25	50	74	99	123	148

APPENDIX D

```
#include <18F452.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use i2c(Master,sda=PIN_C4,scl=PIN_C3)
#byte  sspadd = 0xfc8
main()
{
int16 angle;
byte datah,datal;
sspadd=49;
do{
i2c_start();                    //initiate a new start condition
i2c_write(0xC0);                //device address+write
i2c_write(0x02);                //address of high byte register
i2c_start();                    //restart
i2c_write(0xC1);                //device address+read
datah = i2c_read(1);            //read first byte
datal = i2c_read(0);            //shift second byte
angle = datah << 8;             //shift left
angle+= datal;
i2c_stop();
}while(true);
}
```

APPENDIX E

```
#include <18f452.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#device icd=true
#use delay(clock=20000000)
#byte T0CON = 0xfd5
int16 a;
void main(void) {
set_timer0(0);
T0CON=0xa8;    // start counter0, increment on low to high transition
do{
T0CON=0x08;     // stop counter0,
a=get_timer0();
delay_ms(3000);
}while(true);
}
```

CURRICULUM VITAE

Muhittin ATİLLA was born in 1976 in Polatlı/ANKARA. He was graduated from the University of Eskişehir Osmangazi, Electrical and Electronics Engineering Department in 1999. He was accepted to Istanbul Technical University, Mechatronics Engineering Graduate Program in 2004. He has six years of working experience in Automation and Control Technologies.